本资源来自数缘社区

http://maths.utime.cn:81

欢迎来到数缘社区。本社区是一个高等数学及密码学的技术性论坛，由山东大学数学院研究生创办。在这里您可以尽情的遨游数学的海洋。作为站长，我诚挚的邀请您加入，希望大家能一起支持发展我们的论坛，充实每个版块。把您宝贵的资料与大家一起分享！

数学电子书库

每天都有来源于各类网站的与数学相关的新内容供大家浏览和下载，您既可以点击左键弹出网页在线阅读，又可以点右键选择下载。现在书库中藏书 1000 余本。如果本站没有您急需的电子书，可以发帖说明，我们有专人负责为您寻找您需要的电子书。

密码学论文库

国内首创信息安全专业的密码学论文库，主要收集欧密会（Eurocrypt）、美密会（Crypto）、亚密会（Asiacrypt）等国内外知名论文。现在论文库中收藏论文 4000 余篇（包括论文库版块 700 余篇、论坛顶部菜单“密码学会议论文集”3000 余篇）。如果本站没有您急需的密码学论文，可以发帖说明，我们有专人负责为您寻找您需要的论文。

提示：本站已经收集到 1981－2003 年欧密会、美密会全部论文以及 1997 年－2003 年五大会议全部论文（欧密会、美密会、亚密会、PKC、FSE）。

数学综合讨论区

论坛管理团队及部分会员来源于山东大学数学院七大专业（基础数学、应用数学、运筹学、控制论、计算数学、统计学、信息安全），在数学方面均为思维活跃、成绩优秀的研究生，相信会给您的数学学习带来很大的帮助。

密码学与网络安全

山东大学数学院的信息安全专业师资雄厚，前景广阔，具有密码理论、密码技术与网络安全技术三个研究方向。有一大批博士、硕士及本科生活跃于本论坛。本版块适合从事密码学或网络安全方面学习研究的朋友访问。

网络公式编辑器

数缘社区公式编辑器采用 Latex 语言，适用于任何支持图片格式的论坛或网页。在本论坛编辑好公式后，您可以将自动生成的公式图片的链接直接复制到你要发的帖子里以图片的形式发表。

如果您觉得本站对您的学习和成长有所帮助，请把它添加到您的收藏夹。如果您对本论坛有任何的意见或者建议，请来论坛留下您宝贵的意见。

附录 A：本站电子书库藏书目录
http://maths.utime.cn:81/bbs/dispbbs.asp?boardID=18&ID=2285

附录 B：版权问题
数缘社区所有电子资源均来自网络，版权归原作者所有，本站不承担任何版权责任。

总策划：戴清民
主　编：吴汉平

信息战名著翻译丛书

隐显密码学（第二版）

Disappearing Cryptography Information Hiding: Steganography & Watermarking

Second Edition

[美] Peter Wayner 著
杨力平 严毅 何晓辉 等译
王伟 审校

MORGAN KAUFMANN

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

信息战名著翻译丛书



隐显密码学

（第二版）

Disappearing Cryptography Information Hiding:
Steganography & Watermarking
Second Edition

本书囊括了围绕密码学和水印学竞争在内的各方面技术，这些竞争包括统计学家和密码学家之间的竞争，水印机和自由旋转数字复印的竞争。本书内容易于接受，覆盖知识范围广博，此外，书中包括经过整理的实例、源代码及有关参考文献。

有位评论家开玩笑说，"本书应该是每个恐怖分子的床头必读之物。"然后，他又说，"当然，还适用于所有的自由战士、好莱坞主管、警方官员、首席信息指挥官和需要处处保密的任何人。"谁知道你是不是一个恐怖分子，或许是个自由战士？但本书仅仅谈论技术，而技术是中性的。你，本书的读者，是决定如何使用本书的人—— 你应该用它来阻止恐怖行为、协调一场婚礼、计划一份永久的爱情—— 技术只是中性的，本书只是纸上的方程式，你应该使这些方程式对这个世界具有正面的意义，而不是策划出一个卑鄙的事件。

ISBN 7-5053-8395-7
9 787505 383951

本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书。

责任编辑：郝黎明

ISBN 7-5053-8395-7/TP・4887　定价：32.00元

总 策 划：戴清民
主　　编：吴汉平
书名题字：戴清民

信息战名著翻译丛书

Disappearing Cryptography

Information Hiding: Steganography & Watermarking Second Edition

隐显密码学

（第二版）

〔美〕 Peter Wayner 著

杨力平 严 毅 何晓辉 等译

王 伟 审校

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 提 要

本书描述了怎样利用字词、声音、图像以及怎样在数字数据中把它们隐藏起来，以使使它们看上去像其他的字词、声音、图像。当这些强大的技术被正确地使用时，就能使得跟踪信息发送者和信息接收者几乎成为不可能的事情。这些技术包括加密术（使数据变得不可理解），隐写术（把信息嵌入视频、音频和图形文件），水印术（把数据隐藏在图像和声音文件的噪声里），模仿（给数据穿上“衣服”使得它看上去像其他的数据），还有其他的一些技术。全书共17章，每一章都分为了几个小节，对那些想了解概念而又不想通读技术说明的人来说，每一小节都为他们提供了一个介绍和高水平的概要，并且为那些想自己编写程序的读者提供了更全面的细节。

本书适合信息安全管理人士、管理人员、军方及其他对信息安全感兴趣的读者阅读。

MORGAN KAUFMANN

Copyright©2002 by Elsevier Science (USA). Translation Copyright© 2003 by Publishing House of Electronics Industry. All rights reserved.
本书英文版由美国Elsevier Science公司出版，Elsevier Science公司已将中文版独家版权授予中国电子工业出版社及北京美迪亚电子信息有限公司。未经许可，不得以任何形式和手段复制或抄袭本书内容。

版权贸易合同登记号：01-2002-4132

图书在版编目（CIP）数据

隐显密码学（第二版）/（美）沃纳（Wayner, P.）著；杨力平等译. 一北京：电子工业出版社，2003.2
（信息战名著翻译丛书）
书名原文：Disappearing Cryptography Information Hiding: Steganography & Watermarking Second Edition
ISBN 7-5053-8395-7

Ⅰ. 隐… Ⅱ. ①沃… ②杨… Ⅲ. 密码一理论 Ⅳ. TN918.1

中国版本图书馆CIP数据核字（2002）第104583号

责任编辑：郝黎明
印　　刷：北京天竺颖华印刷厂
出版发行：电子工业出版社 http://www.phei.com.cn
　　　　　北京市海淀区万寿路173信箱　邮编：100036
　　　　　北京市海淀区翠微东里甲2号　邮编：100036
经　　销：各地新华书店
开　　本：787×1092 1/16　印张：18.5　字数：460千字
版　　次：2003年2月第1版　　2003年2月第1次印刷
定　　价：32.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换，若书店售缺，请与本社发行部联系。联系电话：（010）68279077

对隐显密码学的批判和赞扬

在研究隐匿信息的领域里，隐显密码技术是机智的和有趣的。Peter Wayner先生在密码学这个领域就许多技术、应用和研究方向等提出了很直观的观点。本书以涉及知识面广博而使其成为真正的、独一无二的著作。本书是那些想开始研究隐藏信息的人的必读物。

Deepa Kundur博士，多伦多大学

Peter Wayner的《隐显密码学》非常透彻、精确并很有趣，它对语法、代码、研究等方面进行的确定显得特别有意义，而这些语法、代码、研究正是SDMI和RIAA这样的组织赢得“数字千年版权行动”的重要因素。Peter Wayner以易于接受的方式，介绍了隐藏信息的所有主要技术以及对其所进行的攻击。Peter Wayner的这本书很容易会成为标准密码参考书。

Mike Stay, staym@datawest.net

本书第二版为隐藏密码世界和隐藏真实信息科学提供了一个非常现代的、易于接受的介绍。本书囊括了围绕密码学和水印学竞争在内的各方面技术，这些竞争包括统计学家和密码学家之间的竞争，水印机和自由旋转数字复印的竞争。本书易于接受，知识范围广博。它含有经过整理的实例、源代码及有关文献的软件包及其参考附注的评论。

Adam Back博士，Cypherspace Internet Secutity创始人

致　　谢

本书已是第二版，所以，在这里更要感谢广大朋友的帮助。毫无疑问，我还欠所有“加密和编码”邮件列表工作的参与者一份感激，是他们最初的贡献才激励我写了第一本书，他们始终如一的工作热情使得这份邮件列表成为获取信息的最好来源。

一些最新的邮件列表都更多地把焦点放在了这个主题上。“水印”列表和“密码”列表都是以良好的信噪比系数提供高质量的讨论，其他像“**RISKS**”和“**Dave Farber's Interest People**”这些列表帮助提供了一些意想不到的途径。当然，像**Slashdot**，**Kuro5hin**和**InfoAnarchy**这些现代的列表网址是由固定的、适度的讨论来提供的，这些讨论可以帮助信号跳出噪声。对那些在高质量邮件里为我们提交足量的固定信息和深层次想法的社区成员来说，我们不可能根据名字一个一个地道谢。

信息隐藏专题研讨会的组织者通过举办关于这个主题的优秀的研讨会给这片领域带来了学术的严密性。创建、编辑、审查和出版书稿的规程在很多方法上提升了这一艺术形态。由**Springer-Verlag**出版的论文集对那些对这一领域发展有兴趣的人来说，是个很好的资源。

还有其他一些人从其他方面给了我很大的帮助。**Peter Neumann**仔细审查了本书初稿并提出了很好的改进建议。**Bruce Schneier**也非常热心，他从他的第一本书[Sch94]中给了我们文献目录的电子版本。我把它转变为**Bibtex**格式并在这本书里用它作一些参考。另外**Anderson**对文献目录的注解也给了我很大的帮助。

Scott Craver, Frank Hartung, Deepa Kundur, Mike Stay和三个匿名评论者查阅了第二版本。他们的评论帮助我修改了很多错误并提出了很多改善的建议。**DavidMolnar**和**Greg Ruse**也帮助鉴别了第一版的错误。

这本书最初是由**AP PROFESSION**（**Harcourt-Brace**公司的一家分公司）出版的。出版第一版的负责人包括：**Chuck Glaser, Jeff Pepper, Mike Williams, Barbara Northcott, Don Deland, Tom Ryan, Josh Mills, Gael Tannwnbaum**和**Dave Hannon**。

当然，这个新的版本如果没有得到**Morgan Kaufmann**的**Tim Cox**的支持的话，也不可能存在。感谢**Tim**、**Stacie Pierce**和**Howard Severson**对我的帮助以及鼓励。

译 者 序

在翻译本书的过程中，我们一直在想，翻译和出版这本《隐显密码学（第二版）》是很有意义的。书中介绍的用于隐藏信息的隐写术，是一种很独特的、重要而又有趣的信息隐藏技术，它与信息加密技术是不同的。它们两者虽然都能保护通信的秘密，但信息加密技术是明明白白地告诉别人：你传递了秘密的东西。而隐藏信息的隐写术也能传递秘密，并且别人不知道你曾经传递了秘密的东西，这是非常重要的。掌握这种技术，对于保护自己、保护好人，以及打击坏人都是很有作用的。本书的作者在书中也提到"9·11"世贸大厦受恐怖袭击的事件，在恐怖主义猖獗的今天，我们国家的有关部门和有关研究机构，以及信息安全爱好者都应当注重研究、掌握这门技术。

本书与原著一样有三种注解。一种是在一句话的后面标上"①"、"②"等，然后在该页的下端作出具体注解。第二种是在一句话的后面用方括号标出，如"[Blu82]"、"[JJ98a]"等，这是关于参考书的注解，根据它读者可以到本书后部附录中找到具体的参考书目。第三种我们称其为"旁注"，在原著中，它是标在一段话的旁边，即书页的外侧空白处。本书中，将其放在这段话的后面，用楷体字标明是旁注。

由于本书所讲述的是一门比较新的技术，有的术语很不好译。同时，作者还经常以调侃的口吻说话，需要使之转换成适合书面的语言。经过我们的努力，并多方请教有关人士，终于比较准确地把全书翻译出来了。在此，对我们在北京的朋友张宏先生表示感谢，对译典通知讯网（http://www.dreye.com.cn/）、汉英论坛（http://www.overwaters.com/forum/contact.htm）以及论坛上的朋友们表示感谢。此外，读者对书中的阐述如有疑问，可以与我们联系（ylpcn@263.net）。

前　　言

关于第二版的说明

自从本书的第一版出版后，密码学和隐显信息领域在近5年里发生了巨大的变化。来自科学界的兴趣增长了，专注于有关课题的专门会议也增多了，而且，出现了很多新观念、方法和技术，它们中的许多都包含在这本书里。

这种正在萌芽发展的兴趣并不只局限于科学实验室，企业界已经把目光投入到了这个领域，他们希望隐显信息的使用能够给影音和影像的创建者有一个控制结果的机会。这种隐显信息通常被称做水印。隐显有效负载可以包括有关创建者、版权持有者及购买者的信息，甚至包括有关谁被允许享用信息，以及他们隔多长时间被允许推动按钮的专门说明。

虽然私人公司也一直在致力于信息隐藏技术的研究，但有些时候，科学进步的推动并不与一些企业界的愿望要求相调和。科学家们想让有关隐写术运算法则优缺点的报道自由地交流，而一些企业家则害怕这些信息将会被用来攻击他们的系统，所以他们宁愿把这知识隐藏起来。

当记录行业开始聚焦于Scotta A. Craver、John P. Dan S. Wallach、Drew Dean和Edward W. Felten这些人所做的工作时，这种斗争就爆发成为一场公开的战役。这些程序员攻击了许多由SDMI（Secure Digital Music Initiative，安全数码音乐小组）分配的技术，这个组织是由音乐行业的一些成员发起主办的，主要致力于生产水印系统。这些攻击是SDMI在一个旨在测试运算法则力度的公开竞赛上招致的。不幸的是，SDMI的领导者想通过强制那些进入竞赛的人签署一份有关收集奖金的秘密誓约来制约他们。其实，SDMI是试图使想传播他们对公众详细审查结果的人保持沉默，并试图获得公众详细审查的所有政治优势。当这些程序员2001年4月在匹兹堡（美国宾夕法尼亚州西南部城市）的信息隐藏专题研究组上呈出他们的作品时，RIAA（Recording Industry Association of America，美国记录工业协会）发给他们一封信并建议公众讨论这些程序员是否应该被诉讼惩罚。程序员们则在信中加了些自己的方式内容，就是他们称RIAA和音乐业试图抑制他们的第一修改权。这些程序员后来于2001年8月在华盛顿的USENIX商讨会上呈现了他们的作品，很明显，战争的导火线依然存在。一方是希望信息公开共享（即使它可能产生不愉快的结果）的人，另一方是希望建立审查制度从而保持世界安定的人。

这种冲突看上去好像来自于对隐藏信息运算法则的脆弱的察觉。如果有人知道剧本播放机制，他们就可以通过重写或制造不规则的噪声来毁坏信息。记录业一直在担心有人会使用这种怎样破坏SDMI运算法则的知识来摧毁水印信息——当然，这种事很难做。在某些人眼里，惟一的解决办法就是通过禁止交流技术知识来加强安全保护。

这种看法和密码学所持有的观点完全不同。大多数行业认为，公众安全技术是创建安全运算法则的最佳方法。通过隐匿所实现的安全并不像设计良好的运算法则那样成功。结果，

公众安全技术便识别了密码运算法则的许多弱点，并且帮助研究者发展出能够久经考验的解决方法。

一些生产水印工具的公司感觉，除了保密以外，它们别无选择。水印工具还不足以安全到可以抵制攻击，所以，这些公司希望使用一些附加的安全技术来使得它们更加安全。

不幸的是，附加安全技术带来了额外的麻烦。隐藏信息很容易通过压缩、重新格式化和重新记录伪装信息等方法删除。在录音棚、音像商店和打印商店中使用的大多数普通工具都可以删除水印，对此你完全不用多虑。位就是位，信息就是信息，两个事物之间不可能有一成不变的固定的联系。

本书中关于版权持有者和科学家之间的斗争只是一个开始。隐秘的运算法则从来都没有被很久地使用，也没有任何原因来解释它们为什么要被使用下去。与此同时，希望读者能享受这本书带来的所有信息。你可以尽情地享受本书，无论你花多长时间读完它。

使用说明

本书的前几章主要讲的是构成基本的“欺骗包”的要素，例如私钥密码术、秘密共享和错误校正代码。后几章主要描述怎样用多种途径对隐藏信息使用这些技术。每一章都会给读者一个介绍，如果你想加以利用的话，所介绍的这些信息足够你能使用这些数据。

每一章的信息都是按照重点和难点的顺序安排的。对那些不想费力阅读技术细节又想理解概要的人来说，本书每一章的开头都有高水平的内容提要，对那些想通过掌握信息来创建自己的程序的人来说，本书对每套细节都有介绍。对更深层次的、较多数学的细节不感兴趣的人可以跳过每一章的最后一部分，但注意不要遗漏任何重要内容。对采用运算法则有兴趣的程序员则可以钻研一下最后一部分。

所有的章节（除第1章外）都是以讽喻的叙述来试图阐述章节的观点。你可能发现这些叙述有些很有趣，有些很愚蠢，但是希望你能在这些叙述的过程中获得更好的洞察力。在每一章的最后有一个章节小结，列出了这一章的主要重点。

本书的大部分都是以有趣的方式提供信息。但知识就是力量，有力量的人都想提高自己的控制力，所以本书的最后一章就是一些有关政治问题的评论。

目　　录

译者序 ……VIII
前言 ……IX
第1章　概述 ……1
1.1　简介 ……1
1.2　加密的原因 ……2
1.3　它是如何工作的 ……3
1.4　隐写术的用法 ……5
1.5　隐写术受到的攻击 ……6
第2章　加密 ……9
2.1　加密与白噪声 ……10
2.2　信息的测量和加密 ……17
2.3　小结 ……20
第3章　误码校正 ……21
3.1　校正误码 ……21
3.2　创建误码校正代码 ……27
3.3　小结 ……30
第4章　秘密共享 ……32
4.1　分离秘密 ……33
4.2　建立秘密共享方案 ……38
4.3　公钥秘密共享 ……39
4.4　密码文件系统和秘密共享 ……41
4.5　小结 ……43
第5章　压缩 ……44
5.1　模型和压缩 ……44
5.2　建立压缩算法 ……48
5.3　小结 ……53
第6章　基础模仿 ……54
6.1　反向运行 ……55
6.2　仿真的执行 ……59
6.3　小结 ……63
第7章　语法和仿真 ……64
7.1　使用仿真语法 ……65
7.2　创建以语法为基础的仿真 ……71
7.3　小结 ……85

第8章 翻转与反向 87
8.1 反向运行 88
8.2 可逆转机器的建立 94
8.3 小结 100
第9章 噪声中的生活 101
9.1 在噪声中隐藏 102
9.2 位移动 109
9.3 小结 122
第10章 匿名转信器 130
10.1 匿名转信器 131
10.2 转信器的实质 135
10.3 匿名网络 140
10.4 远景展望 143
10.5 小结 143
第11章 秘密广播 144
11.1 秘密发送器 144
11.2 创建一个DC网 147
11.3 小结 149
第12章 密钥 150
12.1 延伸控制 150
12.2 记号算法 152
12.3 公钥算法 153
12.4 零逼近方法 157
12.5 串通控制 160
12.6 小结 161
第13章 排序和重排序 162
13.1 说明 162
13.2 编码强度 163
13.3 恒定形式 164
13.4 标准形式 165
13.5 复合信息的分组 165
13.6 隐藏信息的分类 166
13.7 添加额外数据包 167
13.8 小结 168
第14章 传播 169
14.1 信息传播 170
14.2 数字化 172
14.3 块比较 177
14.4 快速傅里叶解决方法 178

14.5 快速傅里叶变换 181
14.6 用快速傅里叶变换和离散余弦变换隐藏信息 184
14.7 小波 187
14.8 修改 189
14.9 小结 190
第15章 人工合成的世界 191
15.1 创造的世界 192
15.2 文字定位编码和OCR 193
15.3 回波隐藏 195
15.4 小结 196
第16章 水印 197
16.1 嵌入所有权信息 197
16.2 基础水印 200
16.3 平均值水印 202
16.4 小结 203
第17章 密码分析 204
17.1 寻找隐藏信息 204
17.2 典型方法 205
17.3 视觉攻击 206
17.4 结构攻击 208
17.5 统计攻击 209
17.6 小结 211
总结 212
跋 215
附录A Java仿真编码 216
附录B 棒球CFG 246
附录C 可逆语法生成器 258
附录D 软件 268
附录E 更深层次的阅读 271
参考文献 273

第1章 概 述

1.1 简介

大家都知道，计算机里的信息看起来是被完美地定义和确定的，比如银行账目是1 432 442美元还是8.32美元，气温是73°F还是74°F，会议是下午4点开还是在4点半开。计算机仅仅能处理数字，而且数字还必须是准确的。

但生活并不是简单的。信号器和电子器件生产商假装认为数字数据是完美而不可改变的，就像是一个数字的琥珀结晶体一般。但是，自然界充满了噪声，数字只能近似地表现事情的发生。数字信息比自然界提供给它的信息要精确得多。

数字本身就是奇特的东西。所有的数字都能确定地运用算法、方程来进行运算，但同时也能用数值来误导和示错，用欺骗技巧来加密。统计学家称之为数字欺骗。汽车经销商和会计员可以在资产负债表中隐藏违规活动。加密可以只按一个按钮就使成批的数字看起来像是另一组数字。

语言本身常常超过了理性思想的把握。作者围绕着论题摇头晃脑和思考，并依赖细微差别、折射、暗示、隐喻和众多的修饰来表达信息。这些修饰工具没有一样是完美的，人们像是找一个方法辩论“是”那个字的定义。

本书描述了如何利用这些不确定性和不完全性来隐藏信息。本书介绍了如何把字符、声音、图像隐藏到数字数据当中，并使它们看起来像别的字符、声音、图像，把秘密转换成无害的噪声以便使秘密通过网络，并在位流的海洋之中消失。本书描述了如何使数据模拟别的数据来伪装其信号源，并隐藏数据的目的地；如何将会话浸没在噪声的洪流中，以至让别人无法知道会话的存在与否；如何把存在溶化成为虚无的状态，并把它们从虚无之中转换出来，使之再次存在。

传统的密码学以用数学的安全方法来锁住信息的秘密而取得成功。隐藏数据使之不被发现是一个相似的但又有其独特特性的过程，但也常常称之为密码学。这有许多历史性的例子，包括隔室隐藏和机械系统，例如微粒的、或者是瞬时脉冲传输，这将使信息很难被发现。还有别的技术，例如对单词的第一个字母进行编码来伪装内容，使之看起来像别的内容。所有的这些方法都被不断地使用。

数字信息提供了极好的机会，而不仅仅是隐藏信息，还可以为隐藏数据开发一般的理论框架结构。它使得从理论上描述一般的算法，以及描述一个没有密钥的人要想找到数据的难度成为可能。一些算法为自身的强度提供了很好的模型，而另一些并没有。

David Kahn的电码译员为这种技术提供了一个很好的发展历史[Kah67]。

隐藏信息的一些算法使用密钥来控制它们的工作。本书的一些算法用这样一个方法隐藏信息：它有可能再现这些信息而不需要知道密钥。这听起来像是密码学，尽管它是在伪装中隐藏信息。

把这些算法当做隐写术还是密码学？要想在它们之间划清一条界线是武断和危险的。许多很好的加密工具也能产生数据使之像是完美的随机数字，可以说这些工具试图用于欺骗，它像是产生随机噪声来隐藏信息。另一方面，许多密码加密算法是在学习了隐藏信息如何被发现的经验后，才被设计得不容易被直接破解。在一个阵营里面放置一个算法后，时常又忘记它为什么可以在另一个阵营中存在，所以，最好的解决方案就是把本书当做是处理数据的工具集，在提供每一种工具的同时都提供一些指示错误和一些安全性，用户可以结合几种不同的工具来完成自己的目的。

本书出版时的题目是“Disappearing Cryptography”（隐显密码学），这有个简单的原因，因为本书刚面世的时候还很少有人懂得“steganography”（隐写术）这个单词的意思。现在我们仍然保持这个题目，理由是题目并不是购买者靠封面判断书籍的一个方便工具。如果认为这些算法仅仅是隐藏信息的工具那就大错特错了：一些算法提供加密安全的同时，也提供有效的伪装；当一些算法表现出独立性时，另一些则在加密算法有了深层次的理论研究；当一些算法只是基础性的保护时，另一些则是不依靠密钥就极难破解。所以，试图把算法分类为纯粹的隐写术和密码学只是强人所难。

当我们的全部生活都可能是数字信息时，可以设想信息的形式、形状和外表将会有无穷大的量。

1.2 加密的原因

使用本书中的技术可能有好几种理由，其中一些原因是低俗下流的，甚至是麻烦的——我们所称的四骑士——贩毒者、恐怖分子、儿童色情作品作者和洗黑钱者将找到一个方法，就像和利用电话、汽车、飞机、麻醉药、割箱器、小刀、图书馆、摄像机以及其他公用设施一样，为了他们的非法利益而使用这些加密工具和技术。这就不需要解释，为什么这些人能隐藏在匿名和秘密的面纱后面犯下凶残的罪行。

但是，这些工具和技术同样能够保护弱者。在保护本书的利益的同时，下面列出了一些可能好的用法：

1. 你能寻求深层次的个人问题的咨询，例如自杀的想法。
2. 你可以告知同事和朋友有关气味和个人卫生保健问题。
3. 你能安全地约会浪漫的伴侣。
4. 你可以扮演多种角色和以不同的身份开玩笑。
5. 你可以调查工作的可能性而不需要展示当前的工作和可能失去的工作。
6. 当你受到指责的时候，可以向当局匿名检举某人。
7. 你能向报纸杂志揭露不公平和不正当行为的数据。
8. 你可以参加好争吵的政治辩论而不伤害与恰好在辩论另一方的人的友谊。

9. 可以保护你的私人信息不被贩毒者、恐怖分子、儿童色情作品作者和洗黑钱者利用。

10. 警方可以通过秘密工具传递信息以便渗透到坏人之中。

还有许多别的原因，但作者十分惊讶政府官方并没有意识到这些自由在世界上的重要性。许多政府的功能只是幕后交易和权利游戏，匿名通信反映了政治水平的标准。作者常常相信政府摩擦会停止，当一些人想要信息受到约束的时候，信息就能够受到严格的控制，没有人会去做任何工作。他们只需花数小时的时间讨论：谁可以访问这些信息，谁不能访问这些信息。

总结中更详细地检查了这种技术的希望性和危险性。

举例来说，美国中央情报局曾被指责遗漏了前苏联即将解体的信息，他们对国家内讧时发展的苏维埃军队继续发布悲观的评估。一些人将此归咎于贪欲、权力和政治，作者认为应完全归咎于保护信息秘密的无能。情报首脑Bob不能从情报首脑Frad那儿共享秘密数据，因为每样东西都是独立划分的。当人们没有获得新的或可靠的信息，他们就只得仰赖他们的基本成见，这就会出现他们认为苏联是个新生帝国的情形。他们常常对一些问题进行隐蔽的分析，但这样将导致更低的效率。

信息的匿名散播是对社会这个吱吱作响的轮子的润滑剂。只要人们询问了它的有效性，并且认识到它的来源后，就不会乐意跟随在本文之后了，他们每个人都应该能够使用信息的功能了。当匿名信息正确到来时，匿名信息就仅仅是信息。它只是位流，不是子弹，也不是炸弹或者猛烈的攻击。共享信息通常能帮助社会寻求正当的利益。

秘密通信是安全的核心。警察和保安部门不仅仅是用来保护进度表、计划和商业事务的人。本书里的算法就像大门上和汽车上的锁，它将这个权力给每一个人，给每一个人这种权力去保护自己，防卫别人的犯罪和诋毁。这样，警察就不需要去每一个地方，因为人们能够自我保护。

因为这种种原因或者更多的原因，这些算法对保护个人以及私人数据是一个强有力的工具。

1.3　它是如何工作的

隐藏信息的方法有好多种。有一些提供了秘密行动，但并不是所有的这些方法都同样的强固。有些提供给用户一些初学层次的范例式的帮助，有些是采用了大量的自动化处理，还有一些为了安全还能结合特别需求提供多重的保护层次。它们在文件中全都采用一些随机位、一些不确定位和一些未指定位。下面是本书所使用的技术的概要。

- **使用噪声**　最简单的技术就是把信息和噪声放到一幅图片或是一组声音里面，一个数字文件是由一组用来表征光线和声音的强度在时间和空间上精确的点的数字组成。通常，这些数字都被额外精密地计算过，这样才能有效地避免被人所发觉。例如，一张图片上的一个点，可以由220个蓝色单位在总单位从0到255之间的区间内的变化所表征，当这个点由220个蓝色单位转换为219个蓝色单位时，用肉眼将不可能察觉。

如果这个过程被系统地运作，将可以在人眼的感觉极限范围下隐藏大量的信息。一张数字图像有2048×3072个像素，每个像素都包含有24位有关图像颜色的信息。多达756KB的数据可以被隐藏在每一个像素的每一种颜色里至少3位有意义的位之中。这可能比这本书的文字还要多。人类的眼睛不能够察觉这微妙的变化，但是电脑可以把它全部重新构造出来。

- **向外传播信息** 许多复杂的装置能够向外传送含有像素和瞬时声音的文件。无论是人类还是计算机的这种散布，它保护了信息，而且也不容易察觉。许多技术都属于这种类别，工程师首先用这种技术来使无线通信场地减少干扰、抑制干扰和添加一些秘密。现在，要把它们改编成为数字通信并不困难。向外传播信息常常要增加结构的回复力以防止随机的和恶意的威胁。向外传播数据通常这样描述：并不是所有的位都需要重编原始数据，如果一些部分被损坏了，信息仍然能够穿过。许多这样的传播技术把信息隐藏在图片或者声音文件的噪声里面，但这也能被传递其他形式数据所使用。
- **采用统计描绘** 数据常常隐藏在图形中，而计算机则试着通过查看图形来决定数据。例如，在英文文本里，使用字母“p”的时候远远多于使用字母“q”。如果数据能够还原，采用英语语言的统计描述，这样，计算机系统就能够注意到是“p”还是“q”。

很多技术都和产生加密安全随机数字的过程密切相关——那就是说，一个随机数字通量是不能被预测的。一些算法使用数字通量来选择具体位置，而另一些则用隐藏信息把这些随机数值混合在一起，并用信息代替其中一些随机数值。

- **采用结构描绘** 模拟文件类型的统计量仅仅是一个开始。许多复杂的解决方案依赖建立在基础数据的复杂模型中以便更好地摹仿。例如，信息可以隐藏为看起来像棒球比赛的成绩单一样。这些位被隐藏在用来选择的名词、动词和文本的其他部分之间，数据通过文本的分类，以及把单词和位选择配对而得以恢复。这种技术能产生惊人的结果，尽管信息的容量常常看上去是多环的和无方向的，这对于“笨”一点的人和被编程用来扫描特殊字样的电脑来说，已经完全能够解决。
- **随机替代** 许多软件程序使用随机数字发生器来增加场景、声音和游戏的真实感。怪物在数学上的定义看上去更像是随机数字发生器在一张平滑的皮肤上加上斑点、瑕疵、痣、疤痕和沟槽。信息能够隐藏在这些随机数字当中，斑点和疤痕的位置都可以携带信息。
- **次序变换** 一张食品单或许仅仅就只是一张食品单，但那些项目的次序可能携带着大量的惊人的信息。
- **分离信息** 首先声明，这并不是数据需要通过数据包传播的原因。分离信息技术的做法是把数据分成许多数据包通过不同的路径到达目的地。复杂算法能够通过分离信息使n个部分的k个子集被重组为完整的信息。
- **隐藏信息源** 一些算法允许人们广播式地发布信息而不公开他们的身份，即不公开信息的来源。这和隐藏他们的信息本身不同，在某些情况下这是很有价值的。

上述这些不同的技术可以通过不同的方法结合起来应用。首先，信息可以隐藏在序列当中；其次，再隐藏到某些文件的噪声当中；然后，隐藏数据信息源进行广播。

1.4 隐写术的用法

在产品和协议中隐藏数据有多种使用方法。隐藏稍微不同的信息或者结合使用不同的算法可以创建不同用法的不同工具。这里有一些最有趣的应用：

- **改进数据结构** 许多程序员都知道，标准数据结构已经陈旧过时，在新的时期计划外的信息就要既能添加到格式当中又不损坏旧的软件。隐写术就是一个解决方案。举个例子，你添加有关相片的额外信息到相片本身里，这个信息将跟随相片一起传播，但并不干扰那些旧的、不知道信息存在的软件。一些研究者建议了另一个用法：你可以嵌入一段从在后台做X射线数字化的放射线学家的注释。这个文件仍然能在标准工具下使用，节约了医院更新设备的费用。
- **强力水印** 有些数字内容，比如书籍、电影和音像文件的创建者，可能需要添加隐藏信息到文件中，用来描述他们对文件设定的限制。这些信息简单的可能像"This file copyright 2002 by Big Fun"，复杂的可能像"This file can only be played twice before 12/31/2002 unless you purchase three cases of soda and submit their bottle tops for rebate, in which case you get four song plays for every bottle top."

由Ingemar J. Cox，Matthew L. Miller和Jeffrey A. Bloom编写的*Digital Watermarking*是对水印以及对子域（subfield）独特挑战的一个很好的介绍[CMB01]。

一些水印意味着文件经过大量失真后仍能找见。较理想的是水印应该在被修剪、选择、转换和压缩后仍然可以被发觉。惟一真正摧毁水印的途径就是改变文件，使文件不再被识别。

另有一些水印则故意做得尽可能的脆弱，其用途是如果某人想篡改文件，水印则会消失。当文件有可能被篡改的时候，将强力水印和脆弱水印结合应用不失是一个好方法。

- **文件跟踪工具** 隐藏信息可以识别文件的合法用户。如果文件被泄露或散布给未经授权的人，它可以跟踪并将有关情况返回到合法的用户那里。对于作品生产商和使用分级信息的政府机构来说，在每个文件添加单独的标签是一个很有吸引力的主意。
- **文件识别** 包含在文档里的隐藏信息同样包含有数字签名以确保真实性。一个常规软件程序只是显示（或演示）文档，但如果某人需要一些保证，就可以对文件里包含的经过认证标志的数字签名进行核实。
- **秘密通信** 在通信变得不安全后，隐藏信息和保留匿名的软件在政治环境也同样有用，这常常是两个人不能交换信息的时候，因为他们的敌人在监听。许多政府部门仍然浏览网络，进行电子会谈，这正是监视的好机会。在这种情况下，隐藏频道就提供一个政治上脆弱的机会，以逃避控制网络的强权[Sha01]。

隐藏信息的许多使用并不像隐写术和加密学那样分类。任何想要处理旧的数据格式和旧的软件的人都知道，程序员不会总是提供所有文件理想的数据结构。许多基础工具和本书里的隐写工具并没有什么太大的不同。智能一些的程序能包装一些以前从未用过的信息来扩展数据格式，这种程序能比人们对隐写技术所设想的用途产生更多的应用。也许在某一天的某一个地方，拯救一个孩子的生命还要感谢智能数据的操作和隐写术呢！

1.5 隐写术受到的攻击

隐写算法为信息提供保密和安全，可是保密和安全的等级是很难测量的。当一个数据混合进背景之后，什么时候才能有效地消失？评价某个隐写算法的强度的一个方法，就是尝试进行不同的攻击，然后评定算法所抵挡住的攻击。这种方法要达到完美还有很远的距离，但却是最实用的。即使你不断地尝试，也没有任何方法可以预测到所有的攻击。

攻击隐写算法和攻击加密算法非常类似，应用到很多相同的技术。当然，隐写算法允诺了一些保密和安全，更容易受到更多额外的攻击。

下面列出了一些可能受到的攻击：

- **针对文件** 攻击者访问文件并判断里面是否有隐藏信息。这是最弱的攻击方式，但这也正是隐写术起码应抵挡住的攻击的最小极限。许多这种形式的基础攻击依靠对数字图像或声音的统计分析显示文件里面信息的存在。这种类型的攻击对科学来说更多的是技巧，因为人们隐藏信息通过调整统计可以尝试着计算出攻击。
- **文件和原始拷贝** 在某些情况下，攻击者可能需要一份编码信息的拷贝和一份原始的、未编码信息的拷贝。显然，侦测出某些隐藏信息是微不足道的操作，如果两个文件不一样，那么里面肯定含有隐藏信息。真正的问题在于攻击者将对这些数据进行怎样的处理。攻击者可能会销毁隐藏信息，可以通过置换原始信息来达到目的。攻击者也可能尝试分离出信息，或者用他自己的信息来置换。
- **多重编码文件** 攻击者用n种不同的信息获得文件n种不同的拷贝，它们其中的一个可能是或可能不是原始未改变的文件。这种情形可能发生在当公司在把不同的跟踪信息插人到每个文件的时候，攻击者就可以聚集大量不同的版本。如果音乐公司出售有私人化水印的数字音乐文件，那么几个有合法拷贝的歌迷就可以在一起比较彼此的文件。一些攻击者可能会试着破坏跟踪信息，而另一些则试着置换为自己的版本信息。在这种情况下最简单的攻击就是把文件混合在一块，平均化文件的每一个要素，或者创建一个由各个文件提取出的不同部分的混和物。
- **访问文件和算法** 一个理想的隐写加密算法可以经受得起推敲，即使攻击者知道了它的算法。很明显，简单的隐写信息算法不能抵挡这种攻击，任何知道算法的人都能用它提取信息。但是如果你能保持加密算法秘密的一些部分，并且使用密钥来解密信息，就能够抵御这些攻击。本书里的许多算法就是使用密码保护随机数字发生器，从而控制信息是如何隐藏进文件中的。这些随机数字流的作用就像是一把密钥，如果你不知道它，就无法产生随机数字流，也就不能恢复混合隐藏后的信息。

- **完全破坏攻击**　有些人争论说，隐写术并不是特别有用，因为攻击者可以很简单地破坏信息。这当然是真的，在这种情形下，加密学和其他许多协议同样易受到攻击。
- **随机调整攻击**　某些攻击者并不试着测定信息是否确实存在。攻击者可以简单地添加较小的、随机的调整信息到文件中，以期望破坏可能位于某处的信息。在第二次世界大战期间，政府检查员可以在电报中添加微小的变化，期望以此破坏隐蔽通信。这种方法并不是十分有用，因为它牺牲了全部的准确度来压制信息。本书的许多算法能使用对错校验码，从随机改变的一组有限的数字中恢复信息，以抵御这些攻击。
- **添加新信息**　一个简单的方法就是使用相同的软件在文件中编码输入一组新的信息。有些算法容易受到这种攻击，因为它们把隐藏信息的信道重新覆写了。这种攻击仅仅使用随机选择一小部分信道就可以抵抗优良的对错校验码。
- **重新格式化攻击**　这种攻击就是改变文件的格式，因为不同格式的文件存储数据的方式各不相同。现在有许多种不同的图片格式，例如，使用多种的位存储单个的像素。许多基础工具把一种格式转换为另一种格式，帮助画家对付不同格式的图片。因为转换并不是完美的，隐藏信息常常在这个过程被破坏。图片的水印技术常常被设计为抵抗这种攻击，因为世界上的画家们重新格式化图片实在是太普遍了。一个完美的声音水印，例如，在某人用立体声播放音乐后，以及在空气中传播被录音后，水印仍然是可读取的。当然，这些都有限制。重新格式化可以进行彻底破坏，而且很难预料一个文件会遭受什么样子的剪短、旋转、缩放和剪裁。一些最好的算法可以更好地逼近。
- **压缩性攻击**　一种最简单的攻击就是把文件压缩。压缩算法就是要删除文件中无关的信息，而“隐藏信息”通常也被认为相当于“无关的信息”。危险的压缩算法就是所谓的有损压缩，在解压过程中不能正确地还原文件。例如，JPEG图片格式，比起正确还原其原始图片，不如用近似算法好。一些水印算法能抵御大多数压缩算法，但没有一种能抵御所有的压缩算法。惟一一种可以抵御所有压缩算法攻击的隐藏算法目前正在计划和设计中，用于改变一个图像或声音文件的“可视可听的”特征。

不幸的是，隐写术并不是坚固的科学，因为没有一种简单的方法去衡量它工作得有多好。在没人能看见之前，信息必须如何隐藏？在估量这些可能发生的事上，人类的知觉实在是太简单了。

坚固模型的缺乏，意味着建立具有抵御攻击性好的算法是多么的困难。许多算法能够经受得住简单的推敲研究，但抵御不了经过高级训练或者有天分的人用耳朵和眼睛去分析。一些人有所谓的“黄金耳”，可以听出声音文件的变化，而一般人什么都听不见。也许水印对许多购买唱片的人群是完全听不见的，但是如果音乐家能听得出，唱片公司可能就不会再用它。

我们说的坚固模型存在有缺陷，并不是说算法没有实用价值。一个水印只有百分之一的人能听见，而百分之九十九的人没有察觉到。一张有隐藏信息的图片可能被发觉，但这仅仅是一个是否某人正在尝试侦测它的问题。

这里同样有个小小的疑问，水印或者隐写工具并不需要抵御所有的攻击者来取得实质性的价值，水印在经过剪切和基础压缩后，仍然能携带信息给许多人。黑客可能花时间破坏

它，但多数人会将时间用于去做更好的事情。

我们理解的缺陷同样不意味着这些算法不提供某些安全。一些算法内嵌了提供加密强度结构的信息。借用这些想法并使之结合在一起，它们就能够同时提供秘密和安全。

第2章 加 密

纯白色

20世纪的最后一些年里，MegaGoth（市场营销公司）在这个世界上即将倒闭消失的时候，它签署了可能是它的最后一份合同：收购Pinnacle Paint（一家顶尖涂料公司）。这家公司的执行者们疯狂地想到买一些他们从未拥有过的东西来证明自己的存在，他们甚至讨论起用小规模的、私人所有的涂料公司策划他们要统治整个娱乐世界的市场策略。

虽然有人争论说人们是用自己的眼睛来选择颜色的，但是公司执行者们在假设人们能用某种东西识别所买涂料的前提下快速地实行了他们的计划。人们想成为更大的运动机制的一部分，他们不想仅仅为一个房间来选择颜色，人们正在购买着一种全新的生活方式——怎样选择敢于允许自己脱离社会集体的任何生活方式？公司执行者们并不真正相信这种说法，但是却陷入了尴尬的困境，因为他们先前的两个追寻目标已经被MegaGoth所拥有了。幸运的是，他们的老板在批准他们的计划的时候也不知道这一情况，仅仅是律师般敏捷的思维，才把他们从买下所有已经拥有的东西和支付所有的税务的灾难中解救出来。

MegaGoth/Pinnacle Paint的第一步计划是针对不同的人口统计群，用全新的、不同的生产线来生产和重新封装标准的白色涂料。他们的一些计划如下：

- **Moron和Moosehead** 他们是两个活泼可爱的人，如果被强制在艺术班里扩充想像力的话，不知道他们会怎样胡乱涂鸦？Moron可能会选择一头白色的南极奶牛作为他的主题，Moosehead可能会选择画一些在风中毫无目的地飘舞着的雪花。
- **和谐的白色** 白色就是每一种颜色。Star Trek的全体成员称：他们培养的更多新生代将欢迎Bob“亲善大使”在下一个季节来到他们的公司，他的工作就是把其他人的想法感觉投射给他。和谐的白色是为私房屋主服务的，其功能就像为多种颜色混合基准一样。你喜欢蓝色吗？Bob可以接收你的感觉并确认它。你想你的起居室是蓝色的吗？这就需要和谐的白色。你忌妒绿色吗？那和谐的白色也将可以为你服务。
- **苍白色** MegaGoth选择了三个英国的对象并让他们看两部血腥的恐怖电影，然后他们复制这几个人皮肤的颜色（被恐怖电影吓的脸色苍白），并生产出了闻名于世的最纯白色。
- **雪白色** 一种和MegaGoth/Disney（迪斯尼）分公司交换专利权的产品，保证了那些在托儿所的小孩在任何时候都不会感到寂寞。那些白色的墙壁将成为经历魔术电影的另一种方法，而魔术电影是很久以前当迪斯尼还是一家独立的公司的时候生产出来的。
- **使白色变得矮小的白色** Star Trek的成员发现一种白色可以使星星看上去变得矮小，他们花了大量的时间围绕并观察了整个情节。但令人惊奇的是，显示表明真的不能用白色小矮人的身份讨论有关白色矮小星星，倒是可以用超强重力区域的比喻来吸

引人们的注意。现在，每个人都可以通过在自己的墙壁上涂漆这种白色来在相同的比喻里包装隐藏自己。

2.1 加密与白噪声

隐藏信息是狡猾的商业贸易。虽然本书的剩余部分将通过使位看上去像其他的东西来讨论伪装信息，但本书是从检查基础加密为开端的，这是一个很好的想法。诸如AES（先进加密标准）或RSA（由麻省理工学院的Ron Rivest，Adi Standard和Len Adleman合作发展的运算法则）这样的标准加密功能函数，是通过使信息数据变得不可理解的方法来隐藏信息的。他们把信息传送到整个随机事物或是白噪声里面。这种从文件中转移注意力的效果可能不是很好，但这依然是一个很重要的工具。在这本书的后面将描述到：当有很完整的随机数据来源时，很多运算法则和方法都可以执行很好的功能。所以在应用其他的方案之前，应该给文件加密。

密码学领域在第二次世界大战期间就开始尝试生产完美的白噪声了，这是因为Claude Shannon（当时正在为贝尔实验室工作的数学家）建立并发展了信息理论基础，这个基础为实际测量信息提供了理想的框架。

大多数使用计算机的人对一个特殊文件包含有多少信息都有一个大致的看法。比如一个文件中的一个句子，它包含几个空格并且每个空格都占一字节。但大多数人也知道测量信息是难以准确捉摸的，因为如果一个计算机文件的字节数是信息的精确的度量，那么就没有方法能使一个压缩程序能对文件进行压缩。真正的资产是不可能被压缩的，真正的钻石不可能是光滑的，但炸土豆片一直都是装在充满空气的包装袋。这就是它们为什么不以重量而以体积来出售的原因。诸如PKZIP和Stufflt这样的压缩程序的成功，暗示了用字节数来测量文件就像以体积来购买炸土豆片一样有异曲同工之妙。

压缩将在第5章中讨论。

Shannon是根据依赖于概率论的“称重”的方法来测量信息的。他的观点是，如果你能预知内容的话，你就可以有一个包含有大量信息的信件，但如果那些内容很容易预测的话，信件里就只含有少量的信息。洛杉矶（美国城市）的天气预报就不包含很多内容，因为那里经常是阳光明媚的，并且温度都在72°F左右；加勒比海在飓风季节间的天气预报，则包含有大量的可能随时暴发的风暴的潜在信息。

Shannon是用统计概率来测量信息的。一个8位的字节有从00000000到11111111这两个基数的256种不同的值，每一种数值都以相同的概率发生，然后在一个字节里有8位的信息。另一方面，比如像00101110和10010111这两个数值在一个文件里同时出现的话，则每一字节就只含有1位信息量。这两个数值可以被0和1这两个单独的数字所取代，这样整个文件就缩小为原来的八分之一的尺寸。在下文里，我们把一个文件里的信息位数称做平均信息量。

Shannon还提供了测量信息尺寸的精确计算公式（2.2节的主题）。这种信息测量方法为密码学家提供了一些重要的见识。违反编码的数学家依赖深刻地静态分析在文件里搜索模式及特性曲线。在英文里，字母“q”后面的字母一般是“u”，这种模式是一个弱点，它很容易被那些试图到达潜在信息的攻击者们所利用。一个好的加密程序在最后的文件里不应该留

有这样的模式。一个字节的256种位数值都是以平等的概率出现的，这看上去可以塞满所有的信息。

One-time pad（一次性衰减器）是一个加密系统，它就是信息理论背后的基础结构的一个很好的实例。随机数字衰减器是用来产生密钥服务的，但间谍经常使用它：每次使用一张，然后处理掉。所以它就被取名为One-time pad。

一个单独的One-time pad系统可以用一个单独的加密方法来建立。假设某一时刻密码是单独的数字5，并且信息是由大写字母所组成。为了用密码5来给字母“C”加密，则向上计数5个字母到“H”。如果计数要超过字母“Z”，则在字母表的最后回到字母“A”并如此循环下去。用密钥6给字母“Y”加密就变成“E”，解密时反过来就是了。

使用在4.1.1节描述的一次装填延伸，可以把一个密文分成多个部分。

这里有一个加密的例子：

H	E	L	L	O
9	0	2	1	0
Q	E	N	M	O

在这种情况下，密钥就是9、0、2、1、0这五个数字。它们构成了给信息加密的一次一个衰减器。在实践中，这种排序应该是随机的。一个人应该在他或她的大脑里揭示有一些这样的隐藏的短流程①。

Shannon证明了One-time pad是一种不可破损的加密系统，因为最终文件的信息内容与加密钥后的信息内容是相等的，一种很容易说明其真实性的方法就是破坏上面描述的信息“QENMO”。任何一个由五个字母组成的字都可能有潜在的信息，因为它有可能被任何的密钥所加密。比如说“BRUNO”这样一个名字，如果密钥数分别是15、13、19、25和0的话，它就是由“QENMO”所产生的。如果所有的可能性都利用到，那么攻击者就不能使用有关英文的信息或信息本身。排除使用所有的解决方法的信息的平均信息量，本身应该大于或等于密钥的平均信息量。这是理所当然的，因为信息的每一字节和密钥一样都可能有0到256种数值。密钥的平均信息量甚至可能更大，因为信息的数值分布是依靠语言反复无常的变化的，而密钥是可以随机选择的。

一个真正的One-time pad加密系统是不会受大写字母符号所限制的。你可以使用与在一个字节的256种数值使用过的加密处理稍微不同的密码处理方法，一种流行的方法就是使用XOR（exclusive-or，异或：逻辑运算）的操作方法，这种方法是数字世界的简单加法（0+0=0，0+1=1和1+1=0因为它是循环环绕的）。如果One-time pad加密系统是由0到255个数值字节所组成，并且这些数值是以所有可能的方法及途径均匀地分布，那么结果应该是安全的。由于对潜在信息的统计分析可以揭示出密钥，所以pad不能重复被使用，这一点是非常重要的。冷战期间，美国之所以能够读出苏联与其在美国工作的间谍之间的至关重要的通信，就是因为One-time pad加密系统被重新使用起来[Age95]。现在密钥的字符位数比信件里信息的字符位数要少，Shannon对One-time pad是完美的加密系统的证明已经不能再令

①或者由于太多的电视所引起的创造力的限制。

人信服了。

One-time pad是优秀的加密系统，但却是不切实际的。两个想秘密交流的人必须在互相发信件之前很久就安全地安排和交换好One-time pad加密系统，这几乎是不可能的事情。例如，某个人使用网页浏览器给信用卡加密，那么数字在没有交换密码系统的情况下就发送给了贸易商。一般来说，这种加密系统对实践应用显得太庞大了。

很多人都在不断重复地使用pad的同一个部分，来尝试着使处理过程更加有效。如果他们给一个很长的信息加密，他们可以重复使用90210来作为密钥。这样可以使密钥足够小到可以很容易被记住。但是，这很容易引起危险的副本，如果攻击者能够猜到密钥的长度，他们就可以利用这种副本模型，他们可以知道每第五个字母都以同样的数量被转移。找到这个正确的数量值是微不足道的，解决纵横组字的谜团也是很容易的。

2.1.1 DES和现代加密系统

很多不同的加密函数可以很好地完成把信息加密编码嵌进白噪声的工作。今天，一种具有更强实践性和安全可行性的加密运算法则就是由IBM公司于20世纪70年代发展起来的DES。这种系统使用只有56位的密钥信息给64位的数据程序块加密。现今的密钥位数被认为太小了，因为一些计算机科学家他们已经装配出能够在48小时内[Fou98]尝试所有的2的55次方种可能密钥的计算机。

最新的和最有效的取代DES的标准是AES（Advanced Encryption Standard，先进加密标准），一种美国政府通过长期的、公开的争论而选择的运算法则。这种运算法则——Rijndael①，是一个由Joan Daemen和Vincent Rijmen而来的名字，勉强地击败了其他四个具有资格的决赛参加者[DR00，DR01]。

Claude Shannon在他的另外一些作品部分里建议加密应该由两种不同的相互补充的行为所组成：混淆和扩散。混淆由加密信件和用非线性的途径修改信件组成。单独的One-time pad可以混淆每一个字母。扩散包括采用信件的一部分和修改信件的另一部分，以便使最后信件的每一部分都要依赖于很多其他的部分。One-time pad里没有扩散，这是因为密钥的完全随机性使得扩散变得毫无必要。

DES由16种混淆和扩散的交互圆形所组成，每一个数据块里都有64位被加密。它们被平分为两个32位等份。S-box通过其中的一半并使它变得混淆。预先用最佳的方法给数据加密，这是一个真正的随机操作。然后，这些结果以密钥位数组合并用来给另一半加密。因为数据的一半影响到了另一半，所以我们称之为扩散。这种交互圆的模型通常被称做Feistel network。

如果每64位信息数据块都使用不同的S-box，那么交互圆就可以不需要。这样的话，就和One-time pad等效了。但这会使得效率变得很低，因为一个大文件将需要一个相应大的S-boxes。所以对只用64位给信件加密来说，交互圆是一种比较折衷的设计。

混淆和扩散函数的设计是不同的。混淆是故意以尽可能的非线性来构建的。众所周知，线性函数和直线都很容易被预测。鉴于这个原因，选择给DES提供混淆的S-boxes非线性度越高越好。

①Daemen和Rijmen建议这个名词应该这样发音：“Reign Dahl”，“Rain Doll”或“Rhine Dahl。”

创建一个非线性S-box并不是一个容易的过程。原始的技术被分类，这导致了业界对美国政府在设计中安装“通气窗”和“秘密弱点”的怀疑。两个以色列密码学家——Eli Biham和Adi Shamir，在他们最近的工作中显示了S-box里几乎直线性的趋势是怎样被利用，并破坏DES这样的加密系统的。虽然他们的技术很有权威性，并且对像DES这样的加密系统很成功，但是DES的作者还是发现DES本身可以被理想地设计并抵制这种攻击。

另一方面，扩散功能由于技术方面的原因受到限制。理论上，64位块的每一位都会影响到加密的其他位。如果作为某个块的开端的一个位被改变，那么块中的其他每一位都可能以不同的方式被切断。这种不稳定性保证了那些对密码系统的攻击不能准确地定位，因为每一位都影响到了其他的位。

图2.1显示了使用一半函数给另一半加密的过程。一半给另一半加密的交替反复是保证一半的内容影响到另一半的很好的方法。

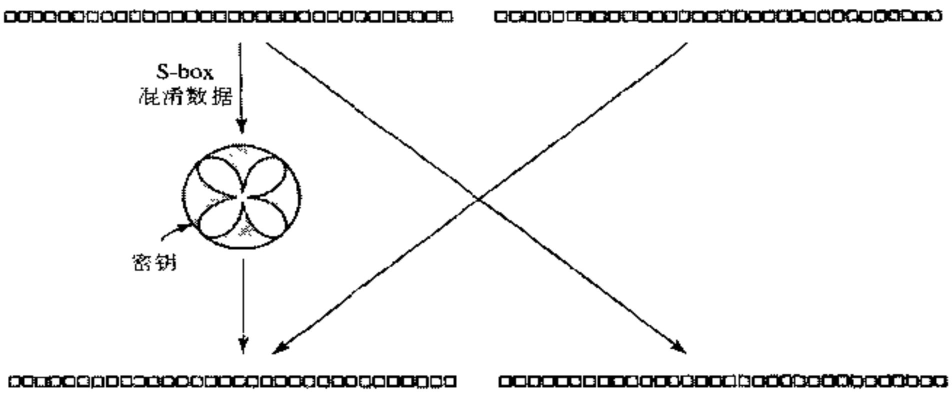


图2.1 一个DES的流程图。64位进入并被分成两个32位等份。左边的一半被使用S-box的密钥加密，其结果与右边的一半相混合，并把这两部分加起来形成了新的左半边。新的右半边只是旧的左半边的复制

DES的扩散甚至更加微妙。虽然一轮过后一半的信息可以影响到另外一半，但平均等份里的位数并没有那么快地相互影响。本书有关这部分的讨论并不涉及到S-box设计的细节，但是，可以说加密数量这一点是被技术所限制了，这里所说的技术是指20世纪70年代中期在加密系统被设计出来的时候所能获得的技术。彻底地扩散信息需要几个循环的过程。

图2.2显示了DES里八个S-box中的一个。这只是一个简单的表格，如果输入S-box的数是000000，那么输出就是1110。这是最简单的加密形式，并且很容易反向。S-box采用6位来作为执行扩散的输入，拥有32位的一半等份被分成了八个4位块，然后每个4位块摄取块中的1位到左边，摄取块中的1位到右边。这就意味着每个4位块都影响到了其相邻4位块的处理。这就可以解释每一等份里的位数是怎样相互影响的。

本书关于这部分已经有很多的详细内容，DES的其余部分对实际需要应用加密系统的程序员来说是很有趣的，其中最重要的课程是DES的设计者怎样选择用一些扩散功能函数对一些混淆功能函数进行隔行扫描，以便生产出不可思议的奇妙结果。

000000 → 1110	000001 → 0100	000010 → 1101	000011 → 0001
000100 → 0010	000101 → 1110	000110 → 1011	000111 → 1000
001000 → 0011	001001 → 1010	001010 → 0110	001011 → 1100
001100 → 0101	001101 → 1001	001110 → 0000	001111 → 0111
010000 → 0000	010001 → 1111	010010 → 0111	010011 → 0100
010100 → 1110	010101 → 0010	010110 → 1101	010111 → 0001
011000 → 1010	011001 → 0110	011010 → 1100	011011 → 1011
011100 → 1001	011101 → 0101	011110 → 0011	011111 → 1000
100000 → 0100	100001 → 0001	100010 → 1110	100011 → 1000
100100 → 1101	100101 → 0110	100110 → 0010	100111 → 1011
101000 → 1111	101001 → 1100	101010 → 1001	101011 → 0111
101100 → 0011	101101 → 1010	101110 → 0101	101111 → 0000
110000 → 1111	110001 → 1100	110010 → 1000	110011 → 0010
110100 → 0100	110101 → 1001	110110 → 0001	110111 → 0111
111000 → 0101	111001 → 1011	111010 → 0011	111011 → 1110
111100 → 1010	111101 → 0000	111110 → 0110	111111 → 1101

图2.2　本图显示了第一个DES S-box是怎样把6位数值传送到4位数值里。请注意一个输入位一般可以改变两个输出位。这种操作是非线性的，它和线性操作执行完全不同的功能

判断像DES这样的加密系统是否强大的最好的办法就是设法破坏它。在较高层次的谈论中讨论像密码破译这样的含有高技术的主题是没有效果的，因为重要的细节通常是如此地微妙，以致于以手织机编织隐喻正好与显著的事实真相擦肩而过。对DES的混淆和扩散的交替层的攻击草图，至少可以给为什么系统是有效的提供一个直接的感觉。

设想你将要破坏DES的一个循环。你有通过一个混淆步骤和一个扩散步骤产生的64位，你想从开端起重新创建64位数，并且决定输入56位数作为密钥。当一次循环结束时，你可以立即发现一半的位数。你拥有主要的优势就是此时DES并没有发生太多的扩散。在每一循环里有32位数是不会被改变的，如果另一半位数是来自于同样的文件，那么就使得做出决定变得更加地容易。另外，这32位数就是反馈到混淆功能函数的那部分，如果混淆过程并不是特别复杂，那么使它反向运转是有可能的。DES的混淆过程非常地简单，并且很容易直接反向。这只需要表格查找，如果你能猜出所输入的密钥或是结构，那么表格查找也就变得非常简单。

现在，如果设想16个混淆和扩散的循环过后你再做上述同样的事情。虽然你可以反向工作，但是，你很快就会发现混淆过程是很难反向运转的。一个循环过后，可以恢复进入功能函数的左半部分的32位数，但是在16个循环过后，你是不可能再得到原始的32位数信息了。如果你试图反向工作，将很快地发现每一件事情都要依赖于其他的事情，这是因为扩散功能导致每一件事都要影响到其他的事。你不能够对你所搜索的一个4位块或是另一个4位块确定出准确的位置，因为所有的输入位数，都影响到了在16个循环的过程中的所有其他的位。这种变化渗透到了整个过程。

Rijndael的主要机理和DES是相似的，但它对现代的CPU（Central Processing Unit，中央处理器）更加有效。DES的S-box对传统芯片的应用相对比较简单，但是对仿真在大多数计算机上使用的通用CPU确实很复杂。AES的混淆是通过一个乘法多项式完成的，并且扩散是发生在信息块的子块被加密的时候。这种数学比复杂的S-box简单了很多，因为通用CPU

就是被设计用来处理基本的算术算法的。

其他四个AES的参与决赛的程序也可能被塞入循环交替的混淆和扩散的模型里。它们都被认为是非常安全的，这就是指它们都提供了更多的“加白”。

2.1.2 公众密钥加密

公众密钥加密系统与DES这些流行的私人加密系统是完全不同的。公众密钥加密系统是依靠实质上不同的，能产生美好的，随机的白噪声的数学分支。尽管这些系统的基础是不同的，但结果却是一样的。

最流行的公众密钥加密系统是Ron Rivest，Adi Shamir和Len Adleman在20世纪70年代当他们还在麻省理工学院时提出的RSA算法（RSA，在数据保密技术中使用的一种通用关键字密码方法，它是基于大数作因子分解的难度而建立的方法）。这种系统使用两种密钥，如果用一个密钥给数据加密，那么只有另一个相应的密钥才可以解开它。可以说第一个密钥是没有价值的，但这不是故障，而是RSA的一种特征。使用RSA时，每个人都可以创建一对密钥，并且可以在电子电话本中公开列出其中一个密钥，即公众密钥，而另一个密钥则必须是保密的，即保密密钥。如果有人想给你发信件，他们可以查询你的公众密钥并用它对发给你的信件加密。目前只有相应的另外一个密钥即保密密钥可以对这封信解密，并且只有你才拥有这个密钥的副本。

在一个相当抽象的理解里，RSA算法是通过在抽象的数学空间里用一个很长很长的圆环安排一套可能的信件来工作的。这个圆环的圆周，我们称之为n，是保密的。你可以把它想像为一串很长的珍珠项链，每一颗珠都代表一个可能的信息。在这个圆环里有数以十亿计的信息，你通过给某个人其中一个珠的指针来发送信件给他（她）。

公众密钥是一个相对大的数字，我们称它为k，一个信件是通过沿着圆环的k个节距一步一步地找到它的位置来给其加密的。被加密的信件就是它所处位置的节距数，保密密钥就是圆环的圆周减去k。从标记被加密信件的数字开始并沿着圆环行进$n-k$个步骤，这样信件就被解密了。因为数字是在圆环中被安排的，所以有可能将你带回到信息开始的地方，回到原始信件。

这串珠的两种性质使得用它来加密变得可能。第一个性质是要有一个被给定的念珠，在整串珠里知道某个珠的准确位置是非常难做到的。如果有一些特殊的第一段念珠是用来为念玫瑰经时确定参考位置而服务的，那么为了确定其中一个念珠的准确位置，就必须计算所有的念珠数。同样的作用在数学里也会发生，必须进行反复的数字相乘来决定一个特别的数字是否是想要的。

这串念珠在隐喻中的第二个性质意思并不是很清楚，但是仍然很容易解释。如果想移动连在一起的k个念珠，几乎可以立刻就跳到那里。这就允许你使用公众密钥系统给信件加密和解密。

这两个特殊的性质非常类似，它们相互也并不矛盾的。第二种说法很容易跳转有任意数量的一串念珠，第一种说法则很难计算出第一个念珠和任意一个特定的念珠之间的珠的数目。如果精于计算，那么就可以使用第二种性质，但不能因此而使用手工计算。

这两种性质的配合使用使得通过跳过大数量的念珠给信件加密和解密变得可能。有些人想破坏系统，因为他们没有通过计算，所以决定不了跳转过程中节距的数目，这两种性质的配合使得这种破坏行为变得不可能。

这个比喻并不十分准确，但是它却抓住了系统的精髓，图2.3阐述了这个道理。

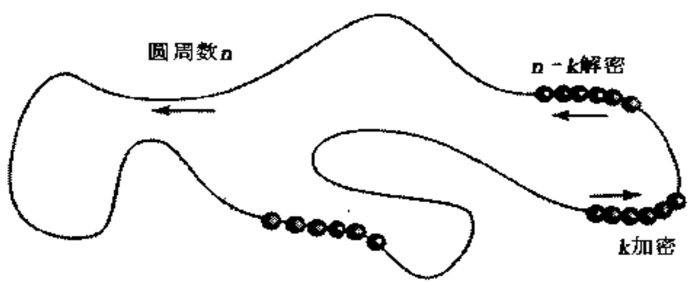


图2.3 RSA加密系统是通过在一个具有秘密圆周的圆环里安排可能的信息来工作的。加密是通过沿着圆环移动一个随机的数目k来完成的。只有拥有者才知道圆周数n，所以可以向下移动$n-k$个节距并覆盖原始信件

在数学上，圆环是通过计算数字的幂并以其他数字为模的方法来建立的。也就是说，圆环里的第一个要素是数字。第二个要素就是数字的平方，第三个是数字的立方，诸如此类。事实上，虽然圆环不只是一维空间，但是主题是一致的。

2.1.3 噪声的随机性

像DES或RSA这样的加密功能函数的输出的随机性如何？不幸的是，这个问题的最佳答案是有关哲学方面的反映："你所说的随机是指什么？"数学很善于从定义明确的问题中找到一致的结果，但是在处理反复无常的行为时，会遇到一些麻烦。

从高层次说明这个问题的最好途径是间接的。如果有一个黑盒子可以看到一个文件的前n位数，并且带点运气地预测到下套位数，那么很明显这个文件就不是完全随机的。有没有一个能够攻击被DES加密的文件的黑箱子呢？最佳答案是没有人了解任何在合理的时间量里工作的黑盒子。一个强制的攻击是有可能的，但这需要大量的机器以及对被加密文件的结构有所了解。所以，可以争论DES和AES的结果应该是随机出现的，因为我们不能成功地预测它们[Way92，Fou98]。

同样的参数也可以放在RSA上。如果有一个黑盒子，它能够摄人数字并且告诉你数字处于圆环的什么位置，那么你就能够破坏RSA。如果输入不是以一个模式集合，那么输出就是很随机的。如果有一些预测RSA的方法，那么这些方法也可以用来破坏RSA。当然，来自于被RSA加密的数值流通量的位数并不是完全地随机，至少在位数层是这样。输出数值都以n为模来计算，所以它们都比n小。因为n不一定是2的乘方，所以有些位在某种意义上更小，这是有可能的。

即使数值不能被预测，仍然可以看到我们想要的东西。比如说，一个被加密的程序可以产生一个没有裂缝但只含有两个数字（像7和11）的结果。这种模式可以是不可理解的、不能预测的，但是你仍然不想为了你的数字掷骰子而使用随机数字生成器。一个立刻能想到

的线索就是如果7和11发生的概率相等，那么这样一个文件的平均信息量很明显就是每数字1位。

很容易创建一个高级参数，使得这个问题不会在DES上再发生，所有可能的输出数值都是以相等的概率产生的。为什么呢？因为DES能够成功地进行解码。有64位输入DES就有64位输出来，每一个可能的输出都只能有一个相匹配的输入，反之亦然。这样每一种可能的输出便产生了。

同样的参数也可以放在RSA上。圆环包含有很多一个个所有的，可能的信件，这些数字以不服从识别力的方法沿着圆环被分配。因此，从功能函数中出来的每一个输出数值都有特别的、相同的出现概率。

虽然这两个参数不能证明从加密功能函数出来的输出是随机的，但它们的确暗示了DES和RSA将通过任何加在其系统上的测试。如果一个测试可以好到足以探测一个模式，那么它无疑将成为破坏编码的一个很好的控制杆。在实践中，简单的测试是支持这些结论的。DES的输出非常随机[①]。很多测试显示"加白"一个随机数字源，使得它更难以处理是一个很好的方法。例如，一些人使用像计算宇宙射线来产生随机数字的随机物理过程来做实验，然而，可能有一个模型是由观测物理学所引起的。消除这种可能性的一个好办法，就是使用DES给随机数据加密并且产生尽可能最白的噪声。

2.2 信息的测量和加密

信息是一个光滑的概念。一个事实有多大？在有健全的概念之前，你必须积累多少数据？这些问题没有一个是很容易回答的，但是在数字数据的帮助下会有一个近似值。Shannon的信息测量方法与概率和随机性紧密地联系在一起。在某种意义上，信息是靠它移动的随机性来定义的。我们的目的就是利用随机性并且用隐藏信息来取代它。能了解我们的目标的尺寸、长度、深度或是宽度是一个很好的开端。

2.2.1 平均信息量

让一个从x_0到x_{n-1}的n个未知数组成的信息通量在以概率为$\rho(x_i)$的通量里发生，这就是Shannon在信息通量里测量平均信息量的方法，也就是说，每个未知数的位数可以写成：

$$\sum_{i=0}^{n-1} \rho(x_i) \log\left(\frac{1}{\rho(x_i)}\right).$$

这个对数是以2为底数的。

如果通量是由0到255个数的字节组成，并且每一个字节数都以256分之一的概率出现，那么通量的平均信息量就是每字节8位。如果只有两个字节，比如说43和95，这两个数每一个都出现一半的时间，另外254个字节根本就不出现，则这个通量的平均信息量就是每字节1位。在这个简单的例子里，位通量是怎样被一个8位数因式压缩成一个未知数的，这应该是

①如果没有使用密钥反馈机制的话，随机性的层次就取决于文件的输入。在DES的某些版本中，一个块结果和这些输入想异或作为下一个块，以便扩散会通过所有的块。如果这不被使用，那么只要模型是以8字节的倍数出现，一些人就可以用这个模型输入一个文件和取出一个文件。

很显而易见的。在更多的复杂的例子里，平均信息量仍然是测量一个简单的压缩算法工作得好坏的方法，Shannon的测量信息法的局限性也是显而易见的。一个在位数0，1，2，...，254，255，0，1...之间无限重复循环的信息通量将出现并包含每字节8位信息，但是真正被传送的信息并不多。可以用大多数计算机语言编写一个能够复制结果的两行短的程序。这种计算机程序能够代替信息通量，而且把这种程序在网络运行，要比发送重复的、永无止境的字节通量的花费实在是便宜许多。

从某种意义上说，这种重复的记录计算机程序是一个很好的信息压缩形式。如果数据是我们以前所讨论过的土豆片，你可能会希望用一个计算机程序的数据线的数目来测量它，这个计算机程序产生的数据就是数据，而不是Shannon平均信息量。还有另外一种叫做Kolomogorov复杂性的测量信息的方法，它是靠确定最小的能够产生数据的程序的尺寸来试图测量信息的。这是一个分析运算法则的理论性很强的工具，但却是完全不实用的。找出最小的程序，这在理论上和实践上都是几乎不可能的，因为没有人能够测试出所有可能的程序。一个程序在C语言程序上可能是最小的，但它在PASCAL语言、Smalltalk语言或者还未编写出来的语言里会是怎样呢？

Shannon的测量信息法在包括了与其相连的未知数时可能就变得更加复杂：

$$\sum_{i,j}\rho(x_i\mid x_j)\log\left(\frac{1}{\rho(x_i\mid x_j)}\right).$$

$\rho(x_i|x_j)$是指信息通量里在x_j发生的情况下x_i发生的概率，总和就是计算所有可能的组合。这种测量方法做了一件好事，就是增加了一些英语语言的特性。一个字母的出现有很大的变化，字母“h”通常是跟在字母“t”而不是跟在字母“q”后面。这种测量方法也增加了一种例如0，1，2，...，255，0，1，...的模式。但是，有很多能由一个简单的计算机程序产生的稍微更复杂的模式，也仍然与第二排序的平均信息量计算相混合。Shannon定义一个通量的平均信息量包括一直到无穷大的所有的排序，计算到这样的高度是不太可能的，因此，较高顺序项在实际运算中就通常会被忽略。计算一个信息通量的第一或第二平均信息量是具有实践意义的，投入在这个运算方案上的空间数目显而易见是压倒一切的。总和里的各项的数字是随着计算的顺序以对数还原（取幂）的方式增长的。Shannon建立了几个评价平均信息量的实验性的方法，但是这种模型的局限性还是非常明显的。

2.2.2 RSA加密

在2.1节，“加密与白噪声”用了长一串念珠的比喻来描述RSA加密，而这部分则是真正的数学角度的介绍。这个系统是由两个互为质数的数字p和q开始的，p和q相乘非常容易，但是如果数字很大的话（比如大概在1024到2048位），就没有人知道一个有效的方法把pq因式分解成为分量p和q。

这是安全系统的基础。如果取一个数x并计算x的连续幂，那么就会有一些数字ϕ使得x的ϕ次方对pq取余（英文：mod）等于x，x对y取余的意思就是x被y除后的剩余数，比如9对7取余等于2，9对3取余等于0，等等。也就是说，如果持续相乘一个x对pq取余后的数字，那么在$\phi+1$步后，它将回到原来的数x。

一个信件被看做数字x被加密。发信件者通过x的e次方给数字x加密，也就是说，计算x

的e次方对pq取余，收件者通过x的d次方给信件解密；就是说，计算（x）对pq取余等于x对pq取余。如果$d \times e = ø$，那么结果就是x。

我们怎样计算$ø$呢？这个数值要取决于计算中的余数、p或pq。有一个被称为Euler Totient的函数$ø(n)$，它的函数值是一个小于n并和n互为质数的整数。如果x是一个质数，那么$ø(n)$就是$n-1$，因为所有小于n的整数都和n互为质数。侥幸地，$ø(n) = ø(n)ø(n)$，所以$ø(pq) = pq - p - q + 1$；例如，$ø(15) = 8$，数字1、2、4、7、8、11、13和14相互都为质数；数值3、5、6、9、10和12则不全是。

如果知道p和q，那么计算函数值$ø(pq)$是很容易的，但是如果不知道p和q，就没有人知道一个有效的方法来计算函数值。这就是RSA运算法则的基础，这串珠子的圆周是$ø(pq)$，沿着圆环移动一颗珠子就相当于x的乘方。

RSA的两个密钥相乘在一起并使得$ø(pq)$的余数为1。有一个密钥是随机选择的，另外一个密钥则是通过找到第一个密钥的倒数并计算得到的。这些e和d在$de = 1$的地方我们称之为余数函数$ø(pq)$。意思就是：

$$x^{ed} \bmod pq = x$$

Neal Koblitz的书[Kob87]为找到这个倒转提供了一个很好介绍。

这可以很容易地转变为加密系统。使用公众密钥加密要计算$x^e \bmod pq$；解密时就把答案在进行d次乘方。就像这样计算：

$$(x^e \bmod pq)^d \bmod pq = x^{de} \bmod pq = x$$

这样就履行了公众密钥加密系统的诺言。有一个密钥e可以被任何公众都可以用来加密一个信件，但没有人能够解密，除非他（她）知道d。而d这个数值则是保持秘密的。

对RSA的最直接的攻击就是找到$ø(pq)$的值。如果你能把pq因式分解成p和q的话，这种攻击就可能变得做得到。没有人知道有比这更好的计算方法了。

实际上实现RSA的加密需要很多细节方面的注意。这里有一些最重要的细节，它们并没有特别的顺序：

- **信息到数字的转变** 数字通常是以字节数来储存的。RSA可以加密任何小于pq的整数，所以必须要有一种固定的方法把一串字节转入或转出为小于pq的整数。最简单的解决办法就是把字节粘合在一起，直到这串字节的数目比pq大，然后移除一个字节并用随机位数填塞取代它，以便使数值比pq小。要把字节转移回去，则只需移除这个填塞就行了。
- **快速取余（即mod）计算** 计算$x^e \bmod pq$并不需要将x进行e次乘方，这将会有抑制性，因为e可能是一个很大的数。一种更容易的解决方法是计算x，$x^2 \bmod pq$，$x^4 \bmod pq$，$x^8 \bmod pq$...也就是说，让x一直乘方下去。然后选择它们中正确的子集相乘得到$x^e \bmod pq$，这个子集很容易确定。如果e的第i位二进制展开式是1，那么就在$x^2 \bmod pq$里相乘成为最后的答案。

[BFHMV84]，[Bri82]，[Mon85]和[QC82]讨论了有效的乘法运算法则。

- **寻找大的质数** RSA系统的安全取决于对pq分解因式的难易程度。如果p和q都是大的质数，那么分解是很难的。如果数值够幸运的话，识别大质数是很容易的，有一些对原来工作状态良好的测试，解决办法就是随机选择一个真正的足够大的奇数并测试看它是否为质数。如果不是的话，就选择另外一个。找到一个质数所花费的时间，其长度接近于一个整数x在其所包含的位数里的大概比率。

  Lehman测试[Leh82]是决定n是否为质数的一种简单的方法。这个测试是这样的，选择一个随机数字a并计算$a^{(n-1)/2} \bmod n$，如果这个数值不是1或－1的话，那么n就不是质数。每一个数值a都至少有50%的机会不是质数。如果重复m次这样的测试，那么我们确定n有1/2次的机会不是质数，但是至今还没有找到能证明这一结论的a，使m＝100是一个很好的开始点。虽然没有绝对的证据，但这确实是个足够好的开始点。

RSA加密是一个用来为公众密钥加密的非常流行的运算法则。很多其他的算法也是可以利用的。对所有算法的不同之处的讨论已超出了本书的范围。Bruce Schneier的书[Sch94]和Gus Simmon的书[Sim93]提供了很好的资料。

2.3 小结

纯数据加密运算法则，是把数据转入到白噪声的最好的方法，它仅仅是在数据中隐藏信息的一个好的方法，例如一些科学家对随机数据进行加密，以使得它变得更随机。加密也是密码学里使用的所有其他运算法则的基础，诸如取出数据块并且在图像和声音文件噪声里把数据块隐藏起来这样的运算法则，需要尽可能接近于随机的数据，这样可以降低信息被发现的机会。

当然，没有什么东西是完美的。有时候太随机的数据可能会很惹人注意，第17章描述了怎样通过寻找比本应该有的随机性更随机的数值来找到隐藏信息。

- **伪装** 好的加密可以使数据进入表面上随机的白噪声。对很多使用随机数据模仿世界的运算法则来说，这是一个很好的开始。
- **安全性** 像Rijndael和其他四个AES的决赛参与者这样的最好、最新的加密算法，没有受到公众化的、实践性的攻击。这些算法在它们抵制攻击的能力方面仅是被设计的，并单独地进行评估。DES仍然是相对比较安全的，虽然相对小的56位密钥难以对付强制性攻击。
- **怎样使用** 加密编码可以从网络上的很多地方下载，见附录D。

第3章 误 码 校 正

正如雪茄与香烟相似而不相等一样:

1. 快速行走（不等于跑步）
2. 美国在线、电脑服务、神奇这三大网站（他们是很大的网站却不提供网络宽带接入）
3. 植物蛋白饼
4. 使用健身器做爬楼梯的动作（并非真正爬楼）
5. 在非正规选美会上获奖
6. 以时速55英里驾驶
7. 在郊区中生活
8. 过新年时发的誓言
9. 像诗歌一样的列表
10. 简单的幽默结构素材的列表
11. 香烟（并不等于雪茄）

3.1 校正误码

停滞不变的计算机原理的基础是：一个二进制位是用“0”或“1”表示。在这个基础下面，仍然是人们生活当中的正常的，稍微随机性的、稍微有点混乱的世界。就像太阳有时候比平时稍微刺眼一点，还有时候一个星期一直在下雨一样。计算机硬件的物理特性有时候也会有一点随机性：有时候硬盘会存储实际上根本不存在的东西，这时硬盘上的瑕疵就是罪魁祸首。有时候外界空间的α粒子可以通过集成芯片发出尖叫般的声音，并因此而改变计算机的运算结果。

计算机设计者设法通过精确的混合与数学计算来聚集所有的随机事物。清洁机用来清除扰乱事物的污物，误码纠正代码的数学方法则用来安排处理稍纵即逝的剩余问题，这种数学计算是数字剖析的可靠的方法之一。数学使得应用稍微松散的态度和方法建立一个数字电路是可能的，但在模拟世界里这种事情永远都不可能发生。设计者们知道，松散可以被巧妙聪明的数学所弥补。

误码纠正代码在很多重要的途径中可以被有效地用来隐藏信息。最显而易见的方法就是用组织好的方式把小错误引入到一个文件里。如果某个人试图使用普通的工具来阅读文件，错误修正量就会修正这些变化（恐怕没有人比这更聪明了）。更多的改进过的工具可以通过用清理过的副本与原文件比较，或者用误码校正原则指出位置的方法来找到这些变化。这些信息就会在错误的地方被编码。

一些音乐CD制作商正在利用计算机的错误纠正机制和CD播放器之间的不同点。有意识地放置的错误将会被CD播放器纠正，但是却会被计算机的机制标注为光盘失败。瞧，这就

是防止CD在计算机中被使用的复制保护。

误码校正代码也可以用来帮助两个人共享同一个通信通道。很多半公众化数据流通量为隐藏信息找到理想的位置。使用某种方法在因特网上的照片和音乐文件里嵌入位数是有可能做得到的，这种方法是抓取所要的文件并用包含有自己信息的副本取代它。在其他人有同样的想法之前这种方法一直很成功。一个信息可以突然地覆盖式地写在另一个信息上，一种理想的解决办法就是重新安排调整信息，使得没有人能够占有像这样的信息通道（哪怕是一小部分）。然后，他们将用误码纠正代码来编写他们的信息。如果两种信息相互作用的话，它们仍然只会相互损坏位数的一小部分，并且误码校正编码将被用来修正它。在许多收音机系统中使用编码也是这种道理。

出现很多数量的误码可能暗示着一个信息的存在。第17章描述了怎样建立一个统计模型以检测像高误码率这样的反常模式。

当然，误码校正代码也可以帮助解决攻击者引入的错误、噪声通道或者格式转化的问题。一些网站减少图像或者应用细微的颜色校正的尺寸，在一些情况下，这些修正只给文件引入很小的变化，并且误码校正代码可以恢复它们。这个额外的力度是很难测量的，因为没有容易的方法预测文件将要受到的损坏。

9.2.6节显示了怎样用随机通道建立一个系统。Ross Anderson，Roger Needham和Adi Shamir使用了一个相似的方法在他们的隐写文件系统[ANS98]中隐藏信息。

有时，把信件分离成很多不同的部分并通过不同的通道运输是很有意义的。从理论上看，如果很少的一部分沿途被损坏的话，信件还是可以被重建的。部分可能消失或被怀有恶意的信使加密。任何这两种情况的一种，误码校正代码都可以预防。

误码校正代码系统加入了许多风味。很多常用的代码具有在一个n位包里携带k位并且在少于m个错误发生时找到正确答案的能力。有很多不同的伴随着不同数值k，n和m发生的代码，但不能让他们空闲，如果有7位并且你想让每一个块都携带至少4位的信息，那么一个标准代码只能校正每块的一个错误。如果想在一个7位块里携带6位信息，那么就不能成功地校正误码，并且只能有50%的机会察觉它们。

比较好的暴露秘密的解决方案可从第4章中找到。

理解误码校正代码的最好的比喻就是想像一下球体。设想一个信件中的每一个字母代表一个球体的中心，有26个球体，每个字母一个，并且它们任何一个都不重叠。你通过发送点在中心的坐标发送一个文件。有时，一个传播的小故障可能会“轻推”坐标使其偏离一些位置。当接收方对信件解码时，如果轻推偏离足够小，他或她仍可以得到正确的原文，这个发送点依旧处于球体的内部。对最好误码校正代码的探索，包括找到把这些包裹球体的最好方法，以便能最好地在一个空间安放球体并传播特性。

虽然数学家是在抽象地讨论球体的包裹和组装，但怎样应用于这个所有东西都是由时开时关的二进制数字组成的数字世界？答案并不是显而易见的。怎样把零点轻推一点位？如果推得足够，它什么时候会变成1？你怎样轻推像252（二进制表示为11111100）这样的数字？显然，一个小的轻推可以把这转变为111111101，即253。但是假如错误出现在第一个位

通过通道的时候会怎么样呢？如果第一位被改变，数字将变为01111100。那就是114，一个128的变化，当然看上去不小。这暗示了球体事实上在一个空间中不能太靠近了。

解决方法是独立地考虑位，并测量具有不同位数的两个数字之间的距离。所以11111100和11111101是区别的一个单元，因为它们只有1位的区别。11111100和01111100也是一样。但01111100和11111101是区别的两个单元。这个距离常被称为“加重平均距离”。

这种测量感觉好像是在一个建筑在像曼哈顿岛（美国纽约一区）方格的城市里找寻两个角落之间的距离一样。第5街和第42街的拐角与第6街和第45街的拐角之间的距离是4块，虽然在曼哈顿岛它们是不同长度的块。你就沿着每一个不同的维数把不同的地方全部加起来。在街道例子里有两个维数：从北到南的街道和从东到西的街道，在数字例子里，每一位置是一个不同的维数，并且8位以上的样本有8维数。

一个误码校正代码的最简单的实例，是使用3位给数据的每一个位编码。这个代码可以校正一个位里的一个错误但不能校正两个位的错误。3位有8种可能的组合：000，001，010，011，100，101，110，111。你可以把它们想像为一个正方体的8个角，如图3.1所示。一个信息0可以被编码为“000”，一个信息1可以被编码为“111”。设想在由“000”转变为“111”的时候发生了一个错误。最近的可能的选择，“000”，是很容易识别的。

误码校正代码和第14章里的频谱传播算法一样，都是用同样的方式使信息在大量位中展开。

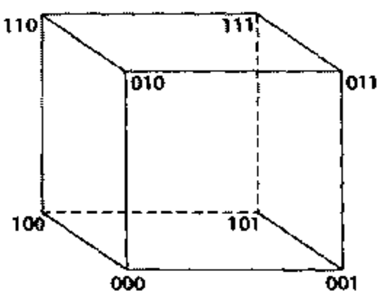


图3.1 8个角的正方体。000和111两个角是用来发送信息0或1的。如果有1位发生了错误，那么可以通过找到最近的角来恢复

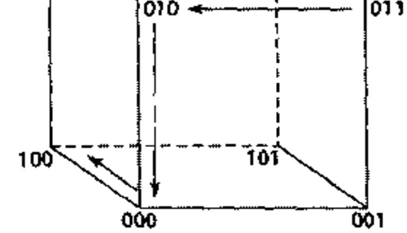


图3.2 加重平均距离显示角“011”离“100”有3步或是一个单元的距离，这是正方体里最长的距离

在“000”的范围中包括至多一个加重平均单元的所有点：001、010和100。两个错误会把一个点推到与其相毗邻的球体。图3.2阐述了在三维空间里加重平均距离的计算。

显然，这个技术可以推广到更高维数的空间里，诀窍就是找到一些能被包装进一个空间的最理想的点。例如，设想一个由点00000、00001、00010...11111组成的五维空间，每一个点都有一个与其相距5单元的的相对点，点00000离11111有5步，点10111离01000有5步。围绕每一个点以2单元为半径创建一个球是很容易的。那就意味着0可以被编码为00000，1可以被编码为11111，一直到发生两个错误并且正确答案被找到。点10110离11111有两个单元，所以它将落到球体内部并且解码为一个1。

一般来说，对于这个简单的方案，奇数维空间要比偶数维空间好得多。设想由000000、000001、000010...111111组成的6维空间，000000和111111都相隔6个单元。但如果画一个绕每个点半径为3的球，那么这些球就会重叠。例如点101010，它离000000和111111都是3个单元的距离，这两个球都包含它。如果试图创建一个使用这种安排的误码校正编码，那么只能在空间里放两个半径为2的球，并且编码只能抵抗每块的两个错误。很显然，五维空间里的5位编码具有更有效的错误抵制能力。

为什么需要每一个空间只包含两个球，这是没有原因的。可以想放很多更小的球，在7维空间里，可以放两个绕任意两个相距7个单元的点为中心，以3为半径的球。但也可以放半径只为1的更大数量的球。例如，可以放置围绕点0000000、0000111、1110000、0011001、1001100、1010001和1000101并以一个单元为半径的球，这些球没有任何一个重叠，并且空间是满的。也可以加一个以1111110为中心的球。这里有8个编码的字词，所以8个不同的消息或3位信息可以贮存在每个由7位编码的字词里面，直到一个位错误被找到并解决。

一般而言，包裹这些更高的维数空间是很难理想地做到的，如图3.3和图3.4所示。很明显，肯定有很多其他的点不在这8个不同的球体之中的任何一个球体。这反映了一个总体上的无效性。

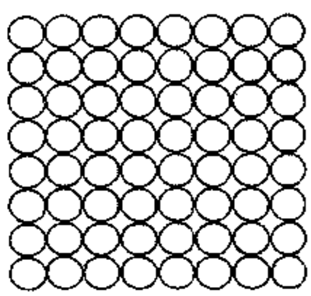


图3.3　这是一个很简略的包裹圆环的方法。如果系统错误不能转变成比球的半径更理想的信号，那么圆环之间的空间就被浪费了。图3.4示出了一种更好的方法

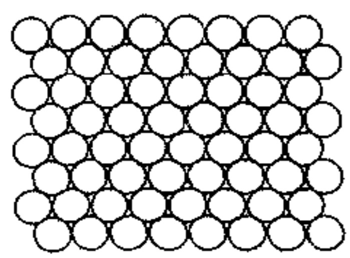


图3.4　一个比图3.3更好的打包圆环的方法，它占原始高度的86.6%，这比一个圆环的半径的一半还宽些

在3.2节中，“创建误码校正代码”描述了怎样建立一般的噪声干扰代码，使用那里提供的算法创建一个包裹4位信息的误码校正代码，或者包裹16个不同信息进一个7位编码的字

词是可能的。那是一个额外位数据，这个编码也可以抵制1位的错误。通过这种方法产生的16个中心如下：

0000000	0001111	0010011	0011100
0100101	0101010	0110110	0111001
1000110	1001001	1010101	1011010
1100011	1101100	1110000	1111111

还有很多其他类型的误码校正代码。包裹球体的比喻是理解基本原理的一个很好的方法，但是对怎样有效地操作却很缺乏指导意义。想像在一个轨道中堆叠、组合球体是很容易，但是想像在复合的多维空间（尤其如果加重平均距离被使用）里怎样做是不可能的。

在实践中，误码校正代码取决于所取数据并加上能被用来恢复数据的额外奇偶校验位的算法。这些奇偶校验位把数据“膨胀”成更多的维数并且把点相互移动开来。例如，在4维空间里点1110和1111正好相互毗邻，但是如果3奇偶校验位被加到每一个点的末尾，结果是1110000和1111111，它们就相隔4个单元了。

如果仔细地看上面的表，第一个4位代表的是四维空间里的所有可能的点，最后的3位是加上去把它推开，形成7维的奇偶校验位。

奇偶校验位的位置在两个不同的代码之间有很大的变化。一些代码可以校正一定数量的、在数据块任何地方都可能出现的误码；另一些代码可以校正发生在脉冲串的误码，它们虽然只能校正一个脉冲的误码，但这个脉冲可以是处于翻转位与上限k之间的任何地方。如果误码不相互连着出现，那么代码就不能修正错误。每一个不同的代码把奇偶校验位放置在不同的排列里，以抓取不同类型的误码。

这本书的剩余部分将依赖误码校正代码给协议加上稳固性，加上随机性，以及提供把信息分离成许多不同的部分的另一种方法。如果信息在通道里偶然碰撞到其他的信息，那么使用误码校正代码就是必要的。

3.1.1 误码校正和白噪声

误码校正代码意味着它能用来校正误码，但它们也可以用于使一个位流量遵照一些模式。一旦一个位集合在误码校正代码里被编码，那么变化就会被引入而不毁坏原有潜在的信息。这些变化加入了随机性，或者有些困难地使数据遵照一个模式。

在实践中，这个方法的最好选择是能够支持大量误码的误码校正代码。一个简单的选择可以是3位误码校正代码传送一位。你把0位写为000，001，010或是100，把1位写为111，110，101或是011。这三个中的任何一个都是可以接受的选择。这样虽然使文件的尺寸增加3倍，但它在重新安排数据结构的时候允许大量的灵活性。

加入随机性是很容易的，但在使数据适应另外一些模式方面还是有局限性。显然，潜在数据必须与这些模式足够接近以便误码能够成功地被引入。从抽象意义上说，模式必须落在球体内部。球体越大，模式工作地越成功。例如，可以很容易地使用以上描述的3位代码来产生1位流量，这1位流在一行里从来没有多于3个1或3个0出现，每1位都可能被编码为以一个1或一个0开始的模式。在另一方面，不能产生这样一个模式——要求在每一行里都有5个1或5个0。

这种技术可以更进一步发展并完全走出误码校正代码的领域。如果正计划使用误码校正代码使你有足够的空间给数据加上随机性，那么将失去很多误码校正的特性。例如，一个翻转变化的位可以把一个代表数值1的二进制数110转变为代表数值0的二进制数010。实际上，你可以完全地忘掉误码校正能力并创建一个代表每一个位的代码清单：1可以被编码为101，010和000。这种方法的主要优点是0和1的分配可以更加平衡。在本节中作为例子的3位代码里，平均有2.25位用来编码一个1，而0.75位用来编码一个0。这意味着文件会有更高的1的百分比。比如，编码后仍然还有一个高的百分比，使用分配给每一个位的随机代码可以移除偏离，但这是本章讨论焦点以外的东西。

3.1.2 误码校正和秘密共享

误码校正代码还有一个功能上的表亲：秘密共享。秘密共享就是一种允许文件被分离成m个部分并且只需$m-k$个部分就可以重新建立文件的算法。很显然，一个可以处理m位里k个误码的误码校正代码能够同样地进行工作，使用这种方法简单地编码文件，然后把m位分裂成m个不同的文件。

第4章将描述秘密共享。

这种方法有一个问题。在一些误码校正方案里，一些位比其他位更有特权。例如，下一章讲的加重平均代码，描述了一个11位加上4奇偶校验位的代码可以校正任何单独的误码。在理想上，一个被这种代码编码的文件能够被分离成15部分，并且任何14部分就足够恢复数据。但是在每个15位块里只有11位数据，另有4个奇偶校验位就是用来校正误码的。如果每一个块的第i位进入第i部分，那么11部分正好足够。密钥就是分配这些位以便使这种情况不再发生。这里有步骤如下：

1. 选择一个提供正确恢复特性的误码校正代码，很容易找到恢复单个误码的加重平均代码。
2. 使用这种技术编码文件。
3. 如果每一块和n个文件里有n位，那么就从每一块里取出1位放入每一个文件。也就是把i位放入文件$i+j \bmod n$。j的选择应该每一个块都不同，它既可以循序地增加也可以通过随机数字发生器来选择。如果一个随机数字发生器被使用，它就应该是一个能够在后来恢复信息中进行再播种的虚假随机数字发生器。

从多数实践的目的来看，误码校正代码并不是共享秘密的理想方法，虽然很容易建立一个能够维持一个误码的加重平均代码，但产生一个每块包含n位并仍然继续存在$n-2$个误码的误码校正代码，却是效率非常低。围绕这个解决方案的理论并没有被优化。

一个更好的秘密共享技术从几何学中直接形成。设想一下，如果想把一个信息编码成一个平面的一点，一种解决方法就是通过这个点画三条直线并把直线分配给不同的人，两条直线就足够重建这个点。通过简单地选择一个代表直线的最大交集的点，这个过程就可以转变成误码校正代码。如果想对更大数量的数据进行编码，可以使用更高维空间并使用平面（二维）或更高维数。这和加重平均代码所做的非常接近，但是在仅有位数被使用的时期，这是很难想像的。

3.2 创建误码校正代码

加重平均代码是简易而优雅的误码校正代码，创建并使用它们是相对比较容易的。可以这样考虑这个创建的问题：即获取引入的信息位并加上允许校正误码的奇偶校验位。实实在在的效应是产生一个只能用一种方法解决的直线方程式的超定集合。

[Ara88]和[LJ83]是这种东西的来源，在那里能找到更多的有关信息。

引入这种算法最容易的方法就是建立一个典型代码，这个代码把11个位和4个新的奇偶校验位进行混合，以便至少1位误码如果出现就可以被校正。输入位是a_1，…，a_{11}，输出位是b_1，…，b_{15}。为了说明这种算法的目的，使用二进制下标更加容易：b_{0001}直到b_{1111}（见表3.1）。

表3.1 一个3位校正代码

输出位	来源
b_{0001}	$a_1 + a_2 + a_4 + a_5 + a_7 + a_9 + a_{11} \bmod 2$
b_{0010}	$a_1 + a_3 + a_4 + a_6 + a_7 + a_{10} + a_{11} \bmod 2$
b_{0011}	a_1
b_{0100}	$a_2 + a_3 + a_4 + a_8 + a_9 + a_{10} + a_{11} \bmod 2$
b_{0101}	a_2
b_{0110}	a_3
b_{0111}	a_4
b_{1000}	$a_5 + a_6 + a_7 + a_8 + a_9 + a_{10} + a_{11} \bmod 2$
b_{1001}	a_5
b_{1010}	a_6
b_{1011}	a_7
b_{1100}	a_8
b_{1101}	a_9
b_{1110}	a_{10}
b_{1111}	a_{11}

说明这一过程最好的方法是利用输出位表。利用一个不同的数，输入位被简单地复制到输出位一侧。如果使用计算机来运行，这一过程就很容易。通过添加不同维数的输入位mod 2，额外的奇偶位被计算出来。它们可以在输出位b_{001}，b_{0010}，b_{0100}和b_{1000}中找到。

误码可以通过计算4个给出误码位置的公式来察觉：

$$c_0 = b_{0001} + b_{0011} + b_{0101} + b_{0111} + b_{1001} + b_{1011} + b_{1101} + b_{1111} \bmod 2$$
$$c_1 = b_{0010} + b_{0011} + b_{0110} + b_{0111} + b_{1010} + b_{1011} + b_{1110} + b_{1111} \bmod 2$$
$$c_2 = b_{0100} + b_{0101} + b_{0110} + b_{0111} + b_{1100} + b_{1101} + b_{1110} + b_{1111} \bmod 2$$
$$c_3 = b_{1000} + b_{1001} + b_{1010} + b_{1011} + b_{1100} + b_{1101} + b_{1110} + b_{1111} \bmod 2$$

这4个方程式得到4位，如果它们被结合成一个单独的数字，就可以揭示出误码的位置。例如，设想位b_{1011}被一个误码翻转。这是引入的位a_7，并且这个位是产生奇偶校验位b_{1000}，b_{0010}和b_{0001}的方程式的部分。这种模型应该是很显然的，奇偶校验位陷在二进制下标只有一个1的储存槽中，一个标准位通过检查下标的二进制数值加到方程式里，如果在位置i有一个1，那么它就会被加进在位置i有一个1的奇偶校验位。位b_{1011}有三个1，所以它在4个方程式里结束。

b_{1011}里的误码效应很容易跟随。位b_{0001}与$b_{0011}+b_{0101}+b_{0111}+b_{1001}+b_{1011}+b_{1101}$的和并不匹配。这意味着$c_0$将计算到1。同样的效应将使$c_1=1$和$c_3=1$。公式$c_2$将保持为0。如果以适当的顺序组合，1011，那么它们就直接指向位b_{1011}。

这些方程式也可以校正发生在奇偶校验位里的误码。如果这些方程式中有一个被翻转，那么只有一个式子产生一个1，其余的方程式将等于0，因为奇偶校验位并不是它们的方程式的一部分。

创建这样一个n位的误码校正代码的一般步骤可以总结如下：

1. 找到最小的使$2^k-k-1<n$的k，这组方程式将编码2^k-k-1位并且产生2^k-1位。
2. 用二进制下标列举输出位：$b_{00\cdots01}$，…，$b_{11\cdots11}$。
3. 奇偶校验位是在下标中只有单数个1的输出位。
4. 把输入位分配给非奇偶校验位，任何顺序都足够，但没有原因不简洁地、不按顺序地执行。
5. 位置i有一个1的奇偶校验位，可以通过把所有在同样的位置i有一个1的输出位（除了奇偶校验位本身）加起来的方法计算，再执行加法取2的模。
6. 为了解码，要计算c_i，它是所有在位置i有一个1的输出位（包含奇偶校验位）之和。如果奇偶校验位相匹配并且不是一个1，就将产生一个0。集合c_i的值将揭示误码的位置，这个代码只能察觉一个误码 。

在这种算法中，k最有效的选择是什么？假设奇偶校验位的数字与输入位数字的对数成比例，把整个文件集中成一个大的块，并只使用小数量的奇偶校验位是诱人的做法。这需要大量的加法运算。在一个n位块里有大约$(n \log n)/2$个加法。较大的块需要更少的奇偶校验位，但是需要更多的计算，它们也只能在一个块里校正一个误码，并且这实质上限制了它们的有效性。最好的权衡办法基于必须携带信息的信道的白噪声。

当加重平均代码每次编码一个字时，这样一个加重平均代码经常可以高速地实现。大多数CPU，包括所有的主流CPU，都有执行指令字的逐位异或操作的指令，指令字通常有32位或64位那么长。异或是对2取模的加法，这种快速的异或操作提供了平行地执行32位或64位编码的一个好方法，它可以使用上面的方程式来完成，但是要用字来代替位运算和用异或操作来代替基本数学运算。

这种方法是一种对误码校正位进行编码的快速方法，但是校正误码则可能很慢。测试误码可以通过看所有c_i的值是否全为0来做到，如果有其中一个c_i不为0，那么代码必须逐个地单步调试每一位并且计算出误码的位置。这样将更加地慢，但是并不比在逐位模式下计算代码慢。

本节里描述的加重平均代码特别优雅，作者看来，正是因为这种方法c_i的值才得以聚集在一起，以找到误码的位置。这只不过是奇偶校验位的一个结果，同样的基本算法可以被使用而不管在其过程中找到什么样顺序的位。任何位b_{0001}通过b_{1111}的置换都可以工作，但恢复的过程并不那么优雅。

这种优雅的排列对以硬件为基础的设备并不是真正的需要，因为误码校正并不需要通过把c_i的值转变为指向误码的索引来实现。为每个寻找完美匹配的位而建立一套逻辑与门是完全有可能的，这意味着奇偶校验位可以被放置在每一个块的开端和结尾，这有可能把它们完全地剥离出去。

3.2.1 周期性代码

在先前的章节里，描述的代码当它只校正每块1位误码，这可能足够了，但是如果块的尺寸很小它就会没有效率。比如，加重平均代码需要3位奇偶校验位来校正4位里的一个误码，从4位里校正出1位，那几乎有50%的损耗。

加重平均代码与理想还相差甚远，因为噪声的本性会损坏数字数据。误码并不是随机分配的，它们通常组合在一个大的脉冲串里，这个脉冲串可能在一个电子振动或一些其他的物理过程干扰后出现。CD-ROM（光盘驱动器）的一个划痕会模糊正好与之相毗邻的几个位，这些误码将损坏任何限制于校正每块1位的加重平均解决方法。

在需要察觉和恢复出现在脉冲串的误码这些场合里，周期性代码是一个更好的解决方法。在这种情况下，奇偶校验位将规则的、有间隔的分配并贯穿在整个位流中。例如，一串位流中每第4个位可以是从前面的一些位计算出来的一个奇偶校验位。因此，如果奇偶校验位数字的每一串位都保持连续，那么奇偶校验位的位置就有很大的变化范围，但是安排它们周期性地出现通常也更加明了。

加重平均代码被设计成与预先确定的位块合作，这里描述的卷积代码将与重叠转动位块合作。同样的卷积技术将与固定的块一起合作，但是为了简洁这里把它省略了。为了避免混淆，这节将使用“子块”这个词来特指被用来建立转动块的更小的一组位。

周期性代码将由一个位子块组成，这个位子块后面跟着一组从前面任意数量的位串中计算出来的奇偶校验位。奇偶校验也可以依赖于后面子块里的一些位，但是由于简洁的需要这个结构也被省略了。

一个卷积代码的一套位看上去可能像这样：

$$b_{(i,1)},\ b_{(i,2)},\ b_{(i,3)},\ b_{(i,4)},\ b_{(i,5)},\ p_{(i,1)}$$

在这里，$b_{(i,1)}$代表子块i里的第一个数据位，$p_{(i,1)}$是第一位奇偶校验位。这个例子里有5位数据位和一位奇偶校验位。

在前面的子块里，奇偶校验位可以是位的任何函数。为了简洁，我们让

$$p_{(i,1)} = b_{(i,1)} + b_{(i-1,2)} + b_{(i-2,3)} + b_{(i-3,4)} + b_{(i-4,5)} \mod 2$$

即每个奇偶校验位都被前面5个子块里的任意一个位所影响。

这些奇偶校验位可以察觉出现在每批转动的5个子块里的具体到5位的一个脉冲串，这意味着如果每两个误码脉冲串在它们之间有至少5个子块，那么误码就将被察觉。误码一旦

被察觉，就会通过要求一个数据的重发来修正。还有一些情况下它也可以被恢复，这将在后面描述。

误码可以通过观察它留在跟随其后的奇偶校验位中的轨迹来被察觉。这种情况下的一个误码脉冲串可以影响跟随其后的任意5位奇偶校验位。当奇偶校验位不相匹配时，前面那批5子块会被重发以修正这个问题。应该很简单地就可以看到怎样展开奇偶校验位，并使得代码系统察觉误码脉冲串是可能的。这些方程式中没有任何一个用来计算依赖于相邻位的奇偶校验位。在这个例子里，在每两个用来计算奇偶校验位的位之间的位流里至少有5位。在加重平均例子里，每一个奇偶校验方程式都依靠于相邻的位。如果因为一个脉冲串的误码噪声使得两个位被翻转，那么误码就会被消除并且是可以恢复的。

用像这个例子一样的简单代码通常是不可能恢复一个奇偶校验误码的。如果有一位奇偶校验位不符合这一实例，那么误码就会被在6个不同位里的一个错误所引入，要找到是哪一个是不可能的。在某种程度上，一个更大的误码脉冲串将使得工作变得更加容易。例如，如果一串位中有三位被翻转，那么三个连续的奇偶校验位也将被翻转。如果奇偶校验位被一个一个单独地检查，那么每一个奇偶校验位都可能被多至6个不同的误码所引起。所以三个误码两两相邻的假设是可以接受的，这样就限制了两个不同点的位置。

例如，这里有一个具有正确奇偶校验位的数据流：

...01010 0 01010 1 1001 1 01111 1 00011 1...

如果前三位被翻转，那么前三位奇偶校验位也被翻转：

...10110 1 01010 0 11001 0 01111 1 00011 1...

每一个单独的误码可能在任意前面的5块中出现，但是代码的重叠特性把误码限制在这里所示的第一块或超过它的两个块的任何一个块里。如果一行里有5位被翻转，那么确切的位置就会被知道并且校正误码是可能的。这把代码推到了一个极限值，并且可以更有希望脉冲串不达到代码察觉误码的能力的极限限制。

本节描述的周期性代码是察觉误码脉冲串的一个好方法，但是它不能帮助校正误码，除非脉冲串到达了其极限值。那里有一些可以利用的信息，但很明显这对恢复数据是不够的。

[Ara88]和[LJ83]都有关于误码校正代码的更多、更好的信息。

3.3 小结

误码校正代码是建立数字系统的最重要的工具的一种。它们允许电子设计师校正那些由于事物本身特性而出现的随机误码，并且给用户提供了能够提高一些数字精密度的方法。如果电子学真的需要提供完美的准确性，那么它们的价格将非常地昂贵。

这些代码对校正出现在发送系统的问题是很有用的。这可能正是我们所期望的，例如，几个人可以使用同一个通道，如果几个人使用随机选择的通道中的一部分，那么代码将校正任何偶然的冲突。

这个领域也幸运地享有许多文献探索的，从许多不同的位置介绍的，在学科范围以外的深奥的东西。不幸的是，本书没有空间把它们全部考虑在内，也没有空间概述不同的方法

对隐写术的或大或小的用处。

第14章描述了像频谱传播一样的隐藏信息应用。这些技术也依赖于把信息分配给文件里数量相对大的单元。如果有几个单元被干扰或损坏，频谱传播解决方法仍然可以恢复信息。

- **伪装** 如果想使用这些代码隐藏信息，最好的解决方法就是干扰一个小的位子集。例如，如果每块都有8位，可以发送每块3位。如果想发送000，那么翻转位0。如果想发送011，就翻转位3，等等。当位在另一端被读取时，误码校正代码将移除误码并且让不经意的读者甚至不知道误码是在那里。还可以使用误码校正代码来恢复它们。当然，这种解决方法换取了隐写术的精确性。偶然的或是有目的的误码将毁坏信息，而用方程式设计的误码校正能力将被用来携带信息。
- **安全性** 误码校正代码在防止人们读取它们这方面并不是完全安全的，位之间的模式很容易被察觉。何况它们对误码还是有很大的抵制性的。
- **怎样使用** 误码校正代码很少直接卖给消费者，但消费者却无时无刻不在使用它们。很多电子设备，像CD播放器和无线电话，都依赖于它们。使用误码校正代码的设计师应该把本书中与此有关的这个巨大而迷人的课题研究透彻。

第4章 秘 密 共 享

三个火枪手中的两个

在Bob曼哈顿岛（美国纽约一区）的起居室里，三个中学时的校友已步入中年，正面临着是否要喝淡味酒（加苏打水的苏格兰威士忌酒）的中年危机。他们都是律师，他们正在被破坏社会公正体系的金钱和贿赂所警醒。所以，在像Batman这样的电影的激励下，他们决定重建“三个火枪手”，并在深夜巡游，寻找那些需要帮助的人。

Bob: 好的，我们明天就将我们的执照存档，以便携带隐藏的武器。星期五，我们去拿货。

Harry: 好，是9毫米的。

Together: 我们每一个人都要去！

Harry: 你知道，我刚刚在想一些事情。我妻子许诺过我们将在星期五去她的表兄家里共进晚餐。她是上个月计划这件事的。我们可以改天去吗？

Bob: 星期天我没有空。我午饭后要去我母亲那里。

Mark: 好的，在上帝的眼里与邪恶斗争没有价值吗？

Bob: 没错。但是我仍然认为我们需要一些偶然性。我们并不是一直都有空的，可能有商业旅行、家庭访问、紧急事件。

Mark: 这还是好的方面，我们有时还有可能受困于交通或者在法庭上受阻。我们需要计划一下。

Harry: 好的，如果我们说“所有有空的只有一个人，而且那天晚上去那里的也只有一个”的话，怎么办？

Mark: 不错，这样更加灵活。但是如果只是我们中的一个人在那里怎么办？

Harry: 有什么不同吗？

Mark: 一个人不可能组成一个真正的团体。他将作为一个义务警员而行动，那天晚上他可以做他想做的任何事，也可能是少一些事而已。

Harry: 所以你需要一个选出的团体？

Mark: 是的。我是说在一些人开始调用三个“火枪手”的名字之前，我们三个中的两个应该到达那里。

Bob: 服装怎么样？如果我们单独的时候我们穿什么？

Mark: 这不要紧。最重要的事是当我们打败敌人的时候我们我们应该叫喊什么，我们呼喊的步调要一致吗？

Together: 三个中的两个组成一队。

4.1　分离秘密

有很多场合你需要把一个密钥或一个秘密分离成许多令人迷惑的部分，以便这个秘密在所有的部分都能使用的情况下才能被复原。这是一个很好的、能够强迫人们一起工作的方法。例如，很多核武器系统，需要两个人同时扭转两个不同的秘密按钮。又如，银行安全存款箱有两把锁，一个钥匙由所有者持有，另一个由银行所持有①。

把信息分离成很多部分是使信息隐藏的一个好方法。每一部分看上去可能都像噪声，但是它们连在一起就产生了信息。这些不同的部分可以走不同的路径，把更深层次的混淆加人到整个过程。

有很多灵巧简洁的方法把一个秘密用数学的方式分离成许多部分。这个秘密可能是一个被加密文件的密钥，或者秘密本身是很重要的仿真陈述。目标就是产生n个能提供重建原始数字的不同的文件或数字。如果有一些原始部分的更小的子集，那么还会有阈值图让你恢复秘密。例如，如果一家公司有五个主管，可能需要3个主管为你提供解开公司用来标识文件的密钥。

这些图的数学计算很简单和直观。第3章显示了错误纠正编码是怎样被用于作为原始秘密共享装置服务的。也就是说，可以通过能纠正错误位的错误，纠正编码来给秘密进行编码以分离它（3.2节的11位编码显示了怎样把一个秘密分离成7个部分，以便如果任何10个部分是可用的时候它可以被恢复）。

这种方法有很多问题。首先，一些位数可能比其他的位数有更大的特权。在表3.1的11位图解里，11位数中的7个持有位信息，另外4个是奇偶校验位。如果正好那7个被放在了一起，那么原始秘密就揭去了它的面纱。如果这些位数之一不见了，就需要奇偶校验位来得到秘密。

其次，允许人们揭开秘密的面纱，即使他们没有持有全部位部分。这可能会有一些冗余。例如，3.1节里描述的3位错误纠正编码即使在3位数中的一个被改变时，也可以恢复正确的答案。这是因为每个位实质上都被转变成其本身的拷贝，如果这3个拷贝被分离成3个部分，那么它们就不能阻止每个人知道秘密，他们手头上都有可能有这个拷贝。他们那部分确实是完整的副本。这个例子有些极端，同样的冗余也可能存在于其他的译本中。

故意加入错误是阻止这种冗余的一个方法。

一种更好的解决办法是使用设计好的运算法则分离秘密，而且除非所有部分的正确数字都可以被利用，否则秘密是不能被恢复的。有很多不同的算法可以完成这种功能，它们中的大多数实际上是几何学的。这种通过图形和图表的算法通常是最容易被理解的。

4.1.1　需要所有部分

本节后面描述的很多算法，可以在只有k部分能被利用的情况下，恢复被分离成n部分的秘密。而很多时候还是需要所有的部分都能被提供。有一些很好的算法，它们在$n = k$时工

①还不清楚为什么银行需要一个它自己的保险箱的钥匙。金库的组合也是为了同样的目的而服务的。

作得很好，但是在$n<k$时却不灵了。在解释其他解决方法之前，先在这里描述一下这些简单的算法。

最直接的方法就是模仿安全存款箱，并使用n层加密。如果$f(k_i, X)$是用密钥k_i加密一个信件X，那么就可以简单地保持秘密并用n个不同密钥的每一个重复地给它加密。也就是说，计算：

$$f(k_1, f(k_2, f(k_3, \ldots, f(k_n, X)\ldots)))$$

每个人获得n个密钥中的一个，并且显而易见除非所有的密钥都被提供，否则秘密是不可能被恢复的。如果有一个密钥不见了，那么一连串都被破坏并且加密层也不能被剥落。

一种更简单的方法就是把秘密设想为一个数字X，并且把它分离成n个部分，这n个部分加起来要等于X，即$X_1+X_2+X_3+\ldots+X_n=X$。如果有一个数字不见了，就不可能确定X的数值是什么。这种解决办法是一次一个衰减器的扩展，它只是很安全而已。拥有$n-1$个部分的人是没有办法猜出那个失去部分的数值的。

在实践中，这种解决方法通常要计算秘密里的每一个位，也就是说，秘密被分离成n个部分。如果所有部分的第一位数被加在一起，它们就可以揭示出秘密的第一位。如果所有不同部分的第二位数被加在一起，结果就是秘密的第二位。这种加法是以二进制数来完成，所以，就可以真正地决定位数里是否有奇数或者位数里1的个数。这里有一个例子：

$$\begin{aligned} X_1 &= 101010100 \\ X_2 &= 101011010 \\ X_3 &= 110010010 \\ X_4 &= 010101100 \\ X_1+X_2+X_2+X_4 &= 100110000 \end{aligned}$$

如果想分离一个秘密，那么你可以先随机产生$n-1$个部分。然后计算X_n使得$X_1+\cdots+X_n=X$。这实际上是很容易的。

$$X_n = X + X_1 + \ldots + X_{n-1}$$

这两种解决方法在安全方面都同等吗？那种加法方法，就是一个以前的衰减器的扩展，是近乎完美的安全。如果不能访问所有的部分的话，是没有任何办法可以破坏系统的。这种方法没有另外的模式。加密层作为安全来说并不需要，有很多不同的加密功能函数和密钥尺寸可供选择，有些选择可能是破坏性的。

另外一种理解分离秘密的方法就是检查方程式的平均信息量，$X_1+X_2+X_3+\ldots+X_2=X$，如果X_i的每一个数值都有m个位，那么就需要有mn平均信息量位数来决定方程式。如果X_i的$n-1$个数值被恢复，仍然有m平均信息量位数或者2的m次方种可能的解决方法可以探索。

直观地，加密方程式：$f(k_1, f(k_2, f(k_2, \ldots, f(k_n, X)\ldots)))$，有同样的性质。如果密钥$K_i$有$m$个位数，那么在方程式里仍然有$mn$个平均信息量位数。不幸的是，函数$f$的复杂性使得提供更加数学性的保证是很困难的。如果基础函数f是足够安全的，足以使用它为基础加密，那么在这种情况下就应该是安全的。但是，如果同样的加密系统被用来使用不同的密钥不断

重复地加密数据的话，对所发生的事就有许多有趣的和不能够回答的问题（一些开始的论文包括[CW93]）。在这种情况下，最简单的方法就是最好的方法。

4.1.2 让一些部分滑动

为什么没有全部中的一部分密钥，也还可以有恢复一些秘密的可能？原因有很多。最简单的算法是基于几何学的基础上的。设想你的秘密是一个数字x。现在选择一个任意的y数值并把它们两个结合在一起，以便在一个平面上表示一个点。为了把这个秘密分离成两个部分，就随机画两条通过这个点的直线，如图4.1所示。

Gus Simmon的书[Sim93]中关于共享秘密的章节是对这个主题的一个很好的介绍。

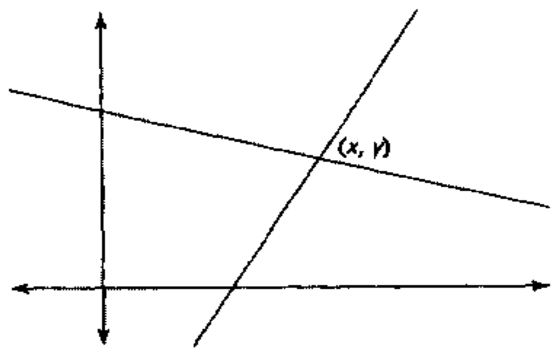


图4.1　一个秘密x，通过两条随机直线在（x,y）的交点被分离成两个部分，其中y是随机的

如果两条直线的交点被找到的话，秘密就可以被恢复了。如果只知道一条直线，就没有人可能知道秘密是什么。

如果有两条直线，那么两部分都需要被用来寻找结果。这个技术可以扩展到*n*个部分，但是任何两个部分都足够恢复秘密。

Roger Dingledine，David Molnar和Michael J. Freedman设计了Free Haven，它使用Michael Rabin的秘密共享算法把一个文件分在多个服务器中。这个系统也提供了一个服务器所有者支付机制[DF00，Rab89]。

随机选择*n*条通过点（*x,y*）的直线。任何一对都将交于点（*x,y*）并允许一些人恢复秘密，就像图4.2显示的那样。

当秘密必须被分离成*n*个部分并且任何*k*个部分必须可用于恢复此秘密时，同样的方法也可以被使用，前提是几何学被扩展到*k*维空间。如果$k=3$，那么直线将被平面代替。三个平面只在一点相交，当两个平面相交时它们会形成一条直线。点（*x,y,z*）将是这条直线上的某一点，但是确定它到底在什么位置是不可能的。

从顶点开始这个过程也是有可能的。这取代了用几条直线的交点隐藏秘密的方法，可以用一条直线隐藏秘密并且沿着这条直线分配所有的点，直线与y轴相遇的地方可能就是秘密，或者可能是直线的斜率隐藏秘密。在这两种情况的任何一种中，知道直线上的两个点，就可以揭示出秘密。图4.3展示了这种方法。

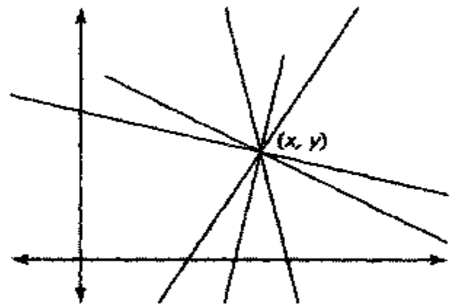


图4.2 一个秘密x，被n个交于点（x,y）的随机直线分离成n个部分。任何两条直线都足以恢复秘密，其中y是随机的

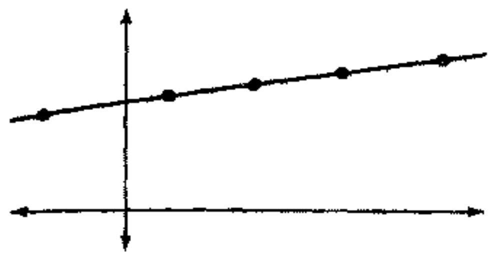


图4.3 这里秘密就是直线本身。沿着直线的随机点被作为秘密的部分被分配，要恢复秘密就必须知道两个点

上述这些信息隐藏系统的每一个方法，都提供了分离秘钥或文件的既好又简单的方法，以致于只需要一小部分人必须在场。应该很容易看到，几何系统在相交点隐藏秘密就像一次一个衰减器一样安全。如果只有一条直线，那么就不可能猜出相交点位于这条直线的什么地方。**X**＝23和**X**＝41 243实际上是一样的。实际上，拥有秘密的一个部分，并没有比未曾拥有该部分给你提供更多的洞察力。在这两种情况下的任何一种中，所知道的只是一些数值x。这就是通常被称做的完美秘密共享系统。

Stephan Brands在他的数字现金支付系统[Bra93]中使用了这个技术（秘密的直线）。

一些人可以尝试抄近路并在交点的x和y坐标里隐藏信息。这好像是可行的，因为可以选择任何一条通过这个点的直线，这实质上改变了系统的参量。如果你有秘密的一个部分，那么就知道一些x和y之间的联系。直线的斜率和y截距正确地描述了x和y是怎样一致地改变的。

在一些情况下，这已足够毁掉系统。例如，设想有一个箱子，它是你为保护一定数量的男人和女人在五月花开的时候逃离英国的藏身之处。在x坐标储藏男人的数量和在y坐标储藏女人的数量是错误的，这种做法泄露了信息：一个英国间谍自然会知道男人和女人的数量可能大致等于社会中男人和女人的百分比，这个额外的信息和一个部分的秘密结合起来，就可以揭示出x和y的近似值[①]。

①你也应该避免把它们作为两个被分成多个部分的秘密分开来储存。在这种情况下，任何这两个秘密的一部分都仍然会产生足够的信息。最好的解决方法就是给这两个数值加密并且将密钥与这个文件分离。

4.1.3　一个更有效的方法

基础秘密共享方法把一个秘密X分离成n个相等的部分，如果秘密有m位长，那么所有的部分也有n位长。这有完全安全的优点。如果一个部分不见了，则只能通过测试消失部分的所有潜在的m位来恢复秘密。

如果因为一些原因使得使用m位很难处理，另一个快速的解决办法就是使用一些函数加密X并把结果分离成n个部分。也就是说，从函数f(X)中取出m位并分配m/n位给每一个部分持有者。

这种方法因为其有效性而牺牲了安全性。取代一个丢失的部分只需要测试代替m位的m/n位的所有可能的组合。但如果这种事有足够的难度，那么这种方法也是可用的。应该注明像f这样的函数必须怎样设计，以便输入任何一位的改变都有可能导致输出位的改变。如果m比一个现代的，像DES或Rijndael这样设计良好的算法的拥塞块小，那这种特性就是很普通的。如果m更大一些，$f_{f(X')}$应该每个位都排序以影响其他的每一个位，当然这是在X'代表X以相反顺序排列的位这一前提下。

4.1.4　否认的本领

本章描述的每一个秘密共享图表都提供了一些在网络上令人难以置信的隐藏信息的机会，为什么一个单独的特殊文件能够足以把信息揭示给发现它的任何一个人，这是没有原因的。把一个文件分离成多个部分是增添否认本领以致其完善的一个理想方法。

第3章描述的误码校正代码也可以被用来添加一些否认本领。

请设想一下，比如最重要的数据储存在一些使用第9章介绍的技术的图像中最不重要的位里，你可以把最重要的数据放在自己的主页背景使用的GIF（可交换的图像文件格式）文件里，然后把它放到网上。这只是你自己的网页，连接它是很自然的。另外一种解决方法就是在网上找到其他三个GIF图像，它们中的一个可能是取自于迪斯尼世界主页，另一个取自于白宫主页，第三个取自于欧洲的阴暗的电脑黑客站点。提取出这些文件的最小的有效位，你虽然无法控制这些位，但是可以使用它们隐藏数据的所有权，这要通过使用这里描述的第一个秘密共享图表来实现。如果把从4个站点复原的数值加起来，最后的信息的出现了。

由Marc Waldman，Aviel D. Rubin和Lorrie Faith Cranor建立的Publius系统使用基本秘密共享算法把密钥分配给一个文件。这个系统发展得很好，它把传统的网络协议统一化以便使它们使用起来尽可能简单[WRC00]。

现在，设想某个字词把信息隐藏在这四个站点的组合里传了出去，这四个当中哪一个是可靠的？迪斯尼世界、白宫、欧洲的电脑黑客？还是你自己的主页背景？使用每个图像中最不重要的位给任何人提供指针方向是不可能的。隐藏信息是这四个的总和，并且这四个中的任何一个可能被操作以保证最后的总和是隐藏信息。是谁这样做的呢？如果能够操作使得隐藏信息在100个可能的图像的总和里才能被找到，那么就没有人会花时间来追踪它了。

这个系统很很像经典书本密码，使用一本书就像一次性便笺。

当然，这个计划也有一些问题。设想如果迪斯尼世界使用了一个来自于像Toy Story这样的电影里的光滑的、射线追踪的图像，这些图像通常都具有连续坡度般的非常平滑的表面，这些表面通常有可以预测到的最小的有效位。这样，肯定就会出现反对上述“操作最小的有效位来发送隐藏信息”的辩论。如果选择如同箔样的金属片的图像，由于它们有非常平滑的表面，所以应该有一个最小有效位的非常“噪杂”的装置。

“Chi Squared”试验以及其它有关随意性的方法可以在Don Knufh的书[Knu81]中找到。

4.2 建立秘密共享方案

秘密共享方案很容易用几何学来解释，但是把它们应用到计算机上则包括了一些折衷办法。最重要的问题是计算机只处理整数，实数范围内的直线既无效又无实践性。例如，在一个为隐藏两条直线的交点的秘密的典型方案中包括五个数字，其中有两个数字描述的是一条直线的斜率和y截距，还有两个数字描述的是第二条直线的斜率和y截距，最后一个数字描述的是交点的x坐标。如果x是整数，那么就不可能随机选择斜率和y截距都是整数的直线。有一些可能是可利用的，但这只是少数情况。

可以使用浮点数字，但是它们加上了自身的不稳定性。第一，必须对数值四舍五入，这是一个很重要的问题，因为两方都必须进行所有同样的四舍五入。第二，可能遇到有很大不同的浮点数学。因为两个不同的CPU对函数x/y可以产生不同的数值，尽管答案可能非常接近，但它们可能是不同的，这是由于不同的CPU使用不同的数值表示方法。很多浮点硬件的用户没有注意这些微小的不同，因为他们所有的计算都是近似值。但这对密码学来说却是一个问题：改变一个加密密钥的一位经常足以使解密失败——即使是最小有效位被改变。

最好的解决方法就是回到整数对质数取余的有限集合。Adi Shamir使用这个范围并从这个范围[Sha79]选择多项式来隐藏秘密，代替了使用直线或平面来隐藏信息。他选择k－1级多项式，$p(x)$，这是第一个参量，$p_0 = p(0)$，用于保藏秘密。参量的每一个点转到每一个部分持有者；k个部分可以用来重建参量并决定秘密，即$p(0)$。

这里是基本步骤：

1. 选择一个质数q。
2. 通过随机选择0至q之间的$k-2$个数值来找到排序为$k-1$的随机多项式$p(x)$，这些将是多项式$p_1 \ldots p_{k-1}$的参量，被保藏起来的秘密就是p_0。
3. $\sum_{i=0}^{k-1} p_i x^i$就是这个多项式。
4. 选择n个点$x_1 \ldots x_n$，计算$p(x_1) \ldots p(x_n)$，这些部分是分配到部分持有者的，k的任何子集足够改变p_0。
5. 恢复数值p_0，使用拉格朗日插值，也就是说，你可以使用这些点来估计多项式在一点的导数。

这个解决方法只使用整数，很显然，应该用k个点来恢复多项式。看待这个问题的最容易的方法就是认识到拥有$k-1$个点并不提供$p(0)$的任何信息。实际上，可以猜测任何潜在的数值$p(0)$，有一些p是可以产生它的。你可以通过采用$k-1$个点，加上你对$p(0)$的猜测来找到

这个p并产生多项式。所以，如果在猜测之间有一对一的映射，那么系统就是完美的。部分持有者并不比行外的猜测者有更大的优势。

当k是一个相当大的数的时候，这个解决方法提供了更大的有效性。在第一节里，几何学的解决方法是产生一个k多项式空间并用$k-1$多项式超平面来填充它。它们的相交k足以揭示出这个点。这个解决方法的问题是超平面会随着k的增大而占用越来越大的空间。作者的意思并不是他们消耗更多的抽象的空间——他们只是需要更大的空间来持有表示他们的信息。这里的Shamir方案需要更大的空间，每一部分仍然是一个放在多项式的点，即（x,y），这实质上更加有效。

4.2.1 制造更多的平等

在本章描述的每一个方案里，秘密被分成了n个部分，每一个部分都是相等的。人类从来不满足于任何像这样的公平——某些人总是想让一些部分比其他的部分具有更多的权利。

满足这个欲望的最直接的方法就是给一些人更多的部分。例如，设想一个方案，需要6个部分来重建秘密，就是说在6多项式空间里你有5多项式超平面的集合。任何6个点的一组足够揭示出秘密，这个秘密是一个核导弹的发射代码的样本。我们说发射导弹需要2个司令官、3个中士和6个士兵来完成，这可以通过把三部分交给司令官、两部分给中士、一部分给每一个士兵来完成。

这个解决方法的一个问题是不同头衔的任意的组合可以结合在一起，即一个司令官、一个中士和一个士兵可以一起工作并揭示出秘密。在有一些情况下这是不允许的。例如，美国国会需要众议院和参议院的大多数来通过一个议案，但是一个议院的选票不能有反对另外议院的看法。所以即使有100个参议员和435个众议员，1个参议也不真正地等于4.35个众议员。一个议案不会被通过是因为有99个参议院支持而只有10个众议员同意。但是可能会有这种情形，一些人通过分配来自于同样共享秘密的部分，给国会两边来产生一个秘密共享方案。

在这种情况下，一个更好的解决方法是把秘密分离成两个相等的部分：X_H和X_s，以便两方都需要用议会的数字签名签署一份议案。然后H_R将被分离成435个部分以便218个部分就足够可以恢复它，H_S被分离成100个部分以便51个部分就足够恢复它。

有大量的组合使这些方案变得可能，尤其是使用秘密的组合和层次可以使任何方案得到实施。复杂系统惟一的问题就是需要很多不同的量纲。例如，如果你想一个使用17个士兵、13个中士和5个将军的系统来发射一些导弹，那么可以使用$17\times13\times5$量纲的系统。这可能有点不可思议，但在数学上确实可能的。

4.3 公钥秘密共享

这一节里所有的算法都共享一套特殊的位数，即秘密。这个秘密可以使用在任何事情上，但它也可能作为解密一些被一个密钥算法加密的信息来使用。分割权利和职责的观念也可以合并成公众密钥（简称公钥）算法。在这些方案里，加密和解密的行为是相互分开并被各自独立地控制。一个密钥的持有者控制加密，另一个密钥的持有者控制解密。如果秘密共享是由共钥加密组合而成的，那么一组必须同意加密一个信件，另一组必须同意解密这个信件。

一个简单的解决方法就是用任意的秘密共享解决方法把任意的公钥算法结合起来。密钥是位的集合，并且这些集合可以使用任意的基础秘密分离法来分离成任意的子集。这种机制非常实用，却也有它的局限性。如果一组人聚在一起加密或解密一个信件，密钥就必须完全地被放在一起。一旦密钥集合在了一起，把它放在一起的人就可以控制它，秘密共享的特点就消失了。

这种方法分裂了由k个人组成的一个小组其解密一个公钥信件的能力，任何人都可以给这组人发送信件，但是这组的所有成员都必须同意才能进行解密。从解密一个信件获得的任何信息不能被用来执行下一步解密，系统依赖于分离的对数强度问题。就是说，假设给出g、p和g^x取余p，是很难求出x的。

私人密钥是由分配给k个控制解密过程的k个数值$\{x_1...x_k\}$组成的。公众密钥是数值$a = g_1^{x_1} g_2^{x_2} ... g_k^{x_k} \mod p$，其中数值$g_i$和$p$是公共可利用数字。$g_i$的数值是这组人定义的对$p$取余发生器，但运算法则将用小于$p$的最随机的数值来计算。

相似的技术也可以被用来产生匿名数字现金和安全投票解决方案[Bra95b, Bra95a]。

一个发给这组人的信件通过选择一个随机数值y被加密计算$g_1^y \mod p$，$g_2^y \mod p$，...，$g_k^y \mod p$。然后，数值$a^y \mod p$被计算出来，并使用像AES这样的算法来产生加密这个信件的密钥。信件由被加密的数据和k个数值$g_1^y \mod p$，$g_2^y \mod p$，...，$g_k^y \mod p$组成。这和Diffie-Hellman变换（另一种公钥加密技术）的精髓相似。

信件可以通过分配k个数值给这个组而被解密。每一个人计算$(g_i^y)^{x_i} \mod p$并把数值返回给这组的领导者，领导者再把它们相乘在一起。

$$a = (g_1^y)^{x_1} (g_2^y)^{x_2} ... (g_n^y)^{x_n} \mod p$$

同样的一套密钥，也可以使用在Diffie-Hellman-Style密钥系统中使用的数字签名，修改译本来给本组产生数字签名。以下是执行步骤：

1. 签名者和签名证实者同意一个挑战性数值c，通常是通过推敲正在被签名的文件来产生的，这也可能是证实者在飞行这样的真正的挑战中生产的。
2. 这组的每一个成员选择一个随机的证明w_i，并计算$g_i^{w_i} \mod p$。
3. 每个组成员计算$r_i = cx_i + w_i$和$g_i^{r_i} \mod p$。
4. 该组删除数值w_i并把其余数值聚集在一个大的集合里为签名服务。这些数值是｛$r_1 = cx_1 + w_1$，...，$r_k = cx_k + w_k$｝，｛$g_1^{r_1} \mod p$，...，$g_k^{r_k} \mod p$｝，和$g_1^{w_1} \mod p$，...，$g_k^{w_k} \mod p$。
5. 任何想检查签名的人可以计算$a^{(-c)} g_1^{r_1}$，...，$g_k^{r_k} \mod p$并确定这和$g_1^{w_1} \mod p$，...，$g_k^{w_k} \mod p$的计算结果相同。

相似的解决方法可以使用RSA类型加密系统来找到。实际上，一些更高级的译本允许RSA密钥在任何一边都不知道因式分解的情况下被产生[BF97, GJKR, BBWBG98, CM98, WS99, FMY98]。

4.4　密码文件系统和秘密共享

秘密共享算法把信息分离成许多部分，以便如果一些人或所有的人可以获得的时候信息才能被恢复。同样的基础代数学也可能被一个人用来隐藏数据，以使得只有他本人才知道真实数据在哪里。如果恰好把信息储存在你的硬盘里并且你又想否认它在那里，则这种应用就很有价值的。

Ross Anderson, Roger Needham和Adi Shamir创建了两个被称为隐写文件系统的译本[AN98]，首次使数学和秘密共享非常相似，所以在这里描述一下。

系统抓取了磁盘空间的一个大拥塞块，并把它随机化，再吸收被密码保护的文件。如果不知道密码，就找不到文件。如果知道密码，然后用随机位产生文件。没有密码的话就没办法确认文件的存在。

他们的方案离完美还很远。为了使方案能够很好地工作，密码必须按层次来分配，这就意味着如果某个人知道一个密码K_i，那么他必须知道所有其他的密码K_j，其中j大于等于0小于i。如果有三个文件，当这个人要访问文件3时，他就必须先访问文件1和文件2。作者想像如果一个人被审问时，他应当有一种办法，只需供出适度的文件密码而不必供出更多的敏感文件。

数学计算就是线性代数学。为了简单，系统都被定义为二进制数，里面的加法是用XOR（异或，符号：⊕）操作，乘法是用AND（与，符号：•）操作。

一个基本的隐写文件系统拥有m个n位长的文件。在一开始，文件就被设置为随着储存在系统的信息的改变而改变的随机数值。这可以帮助把文件系统想像为一个有m行和n列的大距阵，并让C_i代表第i行。

文件j的密码是K_j，一个m位长的向量$K_j(i)$代表第i个位向量。为了从文件系统中恢复出文件j，把所有的行，C_i，$K_j(i)=1$的点相加。就是：

$$\bigoplus_{i=1}^{m} k_j(i)C_i$$

怎样在一个系统里储存一个文件的？这里有储存一个文件的简单步骤的基本策略：

1. 把文件分离为n个位块。
2. 给每一个块选择一个密码。一个好的解决方法就是连接一个随机的串S，在输入密码之前，用一个密码安全混编函数H给文件混编，并把第一个$m-1$位作为K_j对待。
3. 在K_j上加一个奇偶位以确信它是正确的长度。使用奇数奇偶校验保证向量中的1位数是奇数，就是说，如果在第一个$m-1$位里有很多个1，那么就把最后的一个位设为1；如果有一个奇数则设为0。
4. 用分离的加密函数给块加密，理想的情况是使用一个通过添加不同的随机串而创建的不同的密钥。文件系统的直线性决定了加密函数是非常必要的。如果一个攻击者可以建立文件的某一部分D，那么就可以解决一些直线方程式来恢复K_j。如果文件没有被加密，并且攻击者可以猜出一个短语及它在文件里的位置，那么就可以检出这个文件。

5. 对$K_j(i)=1$的所有i，用$D\oplus C_i$来代替C_i。

基本算法将在系统里储存一个文件，因此，这个算法如果被重复使用的话，可能会出现一些问题，因为文件可以重写其他的文件。如果想在系统中储存多于一个的文件，你需要保证它们不会互相破坏。

最简单的解决方法就是选择K_j的m个数值以便它们是正交向量。也就是当且仅当$i=j$时，$K_i \cdot K_j=1$；其他情况则$K_i \cdot K_j=0$。如果密码向量是正交的，那么m个不同的文件可以在不相互打扰的情况下储存在系统里。

换句话说，任何时候D的数值与不同的行C_j相加，D的数值都不会使另外的文件失真。为什么呢？如果$K_p \cdot K_q=0$，那么K_p和K_q都有很多特定数量的值为1的位，想像一下你正在对所有$K_p(j)=1$的j用$D\oplus C_j$代替C_j，这些行中的一些也有可能储存被密钥K_q定义的其他文件的部分。为什么这个文件不被干扰？因为D将只被加在特定数量的行，且数值会被删除。$D\oplus D=0$。

请参考一下这个有关K的例子：

0	1	1	1	0
1	0	1	1	0
0	0	1	1	1
1	1	1	0	1
1	1	0	1	1

这个距阵包括5个文件的密钥。例如，第一行规定第一个文件由$C_2\oplus C_3\oplus C_4$组成，第二行指明第二个文件由$C_1\oplus C_3\oplus C_4$组成。如果新的数据被储存在第一个文件里，那么C_2，C_3和C4将分别变为$C_2\oplus D$，C3$\oplus D$和$C_4\oplus D$。如果是发生在第二个文件将会怎样变呢？$C_1\oplus C_3\oplus C_4$会变为$C_1\oplus(C_3\oplus D)\oplus(C_4\oplus D)=C_1\oplus C_3\oplus C_4\oplus D\oplus D=C_1\oplus C_3\oplus C_4$。

这里有一个从m密码的列表中建立K的基本算法，P_1，P_2，…，P_m，重复每一个密码P_i。

1. 让$K_i=H(P_i)$，这里H是某种密码安全噪声干扰函数。
2. 对所有的$j<i$，让$K_i=(K_i \cdot K_j)\ K_j\oplus K_i$。这个规范的正交集删除了与新行重叠的、先前的向量的一部分，即它保证了在新行和旧行这两行里，只有特定数量的数值为1的位。对所有先前的数值它都是这样做的。
3. 如果$K_i=0$，那么一个错误就发生了。新的被选择的密钥并不与先前的密钥相独立，设$K_i=H(H(P_i))$并重新试一次。继续噪声干扰密码直到一个可以接受的数值K_i被找到，这个数值和以前的密钥是规范正交的。

这个算法并不相互独立地计算数值K_i，必须知道所有j小于i的K_j的数值以计算K_i。这比理想还差一点，但目前这是不可避免的。Anderson，Needham和Shamir决定通过铸造一个直线文件访问层来把限制转变成一个特点。如果能读文件i，就能读所有文件j，在这里，j是小于i的。

把所有的密钥都强制成一个分等级的顺序并不完全是人们所希望的。另外一种找到密钥的技术就是把每个人限制在一个特定的子空间里，也就是把密钥空间分离成规范正交部分。如果"i"这个人想选择一个特殊的部分K_i，那么那人就必须检查一下，看那个K_i是否在其正确的子空间里。

把密钥分离成规范正交子空间的最简单的方法就是把密钥里特定的位强制设为0，Alice可能使用的是前十位被设为1的密钥，Bob可能使用的是第二个十位被设为0的密钥，如此等等。

如果需要的话，Alice，Bob和其余的人可以同意一个随机的旋转距阵R，并使用它来旋转子空间。这样如果在所有正确的空间RK_i都是0，Alice就只能选择一个密钥向量。

这个文件系统的译本也有一点稍微难使用。如果想读或写一个文件D，那么可能必须访问m个之多的其他的行。如果m很大的话，这个因素就是实质性的，这可以通过使用非二进制数值代替K_i个别要素的位来减少。

4.5　小结

秘密共享是一个干扰通过网络的文件，使得没有人能找到它们的理想的方法。这也是一个人们否认责任的很好的方法，在一些情况下，有些秘密部分可能是来自于与即将到来的麻烦无关的人的网页。

- **伪装**　就是秘密共享也让你共享责备。
- **安全性**　这里的算法反对来自于拥有少于必需部分的人的攻击，它是无条件安全的。
- **怎样使用**　这里描述的异或算法是很容易实现的，它创造了一个分离信息的理想的方法，每一部分都必须呈现，并能把信息放回在一起。

第5章 压缩

电视节目表

晚上8:00频道2 IRS（美国国税局）贝弗利市“Hills Model Patrol”。新的润唇膏介绍。

频道5（QUS） 现场气笛风琴演奏：音乐的侦探。一个军官之死。

频道9（PVC） 北部巡警。城镇委员会禁止在城镇会议上的古怪行为，但时间不会很久。

频道14（TTV） Def N. B. 贝多芬对女王的指责。

晚上9:00频道2（IRS） 超级英雄帮。由于多种原因，邪恶正慢慢返回。

频道5（QUS） 深呼吸教育节目。该节目在顶级狗食厂发现了有毒的魔物。

频道9（PVC） 妈妈的一个偷盗癖。像妈妈剽窃女儿的英语试卷这样的家庭重点话题。

频道14（TTV） 简单的干酪。消费者对3重凤尾鱼比萨饼的要求。

晚上10:00频道2（IRS）X知道最好。外来的继母证实了爱是为世俗利益所束缚的。

频道5（QUS） Dum De Dum Dum。侦探笨蛋遇上了谋杀出版人案件。

频道9（PVC） 背叛区。Bob背叛了Jane。

频道14（TTV） 贝弗利山宇航员。Buzz发现在空间中没有林荫小道。

5.1 模型和压缩

生活经常可以归纳为公式。至少在电视上是这样，电视里解决方法每30或60分钟就出现。当你知道公式的时候，就很容易归纳出信息并压缩它。据说一个网络工程师委托电视并通过给演出人一两个词的备忘录“MTV（音乐电视）偷窃”来说明迈阿密市的恶习。你可以用一个简单的句子“跟随宇航员的那个”，很容易具体说明Gilligan岛的一个情节。任何看过一个情节的人都将知道一些宇航员出现在岛屿上，提供了一些人们将被营救的希望，但是当Gilligan把事情振作起来时，这希望在最后还是破灭了。

压缩普通的信息就只是找到能描述数据的正确的公式。很容易找到一个工作得相当好的简单公式，但是证实一个能够很好地压缩数据的公式却是令人发狂地难。找到一个为像书本或电视这样说明类型的工作的一个好公式通常有经济价值。人们总是在寻找把他们的数据库和通信或本分开的好方法。

对任何想隐藏信息的人来说，压缩数据可以引起他们的兴趣。原因有几个：

- **更少的数据更容易地被处理** 这说的是它本身。基础文本可以很容易地被压缩达50%～70%，图像可以压缩90%。

- **被压缩的数据通常变得更白**　压缩不应该损坏信号里的信息。这意味着如果文件的尺寸减小那么每位信息就应该增加，每位更多的信息通常显示出更大随机性。

测量信息的详细内容可以在2.2节里找到。

- **反向压缩可以模仿数据**　压缩算法试图找到一个适合数据的公式，然后作为压缩后的数据返回公式的具体细节。如果将随机数据输入到一个压缩函数中，就会输出一个符合公式的数据。
- **压缩算法辨识噪声**　好的压缩算法懂得人们的感知是怎样工作的，像JPEG和MP3这样的算法可以从一个文件中去掉不被用户注意的信息。这些信息可能会被密码学家利用来查找位置，位置中重要数量的噪声可以被一个隐藏信息所代替。

9.2.7节显示了JPEG算法是怎样确认一个图像里有多少空间被利用的。

当然，这种方法也非常危险。如果JPEG算法去掉了一些信息，那么在这个位置插入的任何信息都有可能被去掉。一个攻击者或是一个善意的程序员沿着路径可以压缩一个文件以节省空间和毁坏信息。这使得该技术对像受到水印这样的脆弱保护的数据变得很危险，但是如果能够有理由保证文件不会沿着路径被压缩，那还是有潜在的用途的[①]。

14.6节里的一个水印学算法故意在图像的最高有效位里隐藏信息，以避免在压缩过程中这些信息被损坏。

因为这些原因，所以压缩是一个很重要的工具。很多好的商业压缩程序因为第一个原因就已经存在了，而很多好的加密程序使用压缩作为一个额外的能力资源。第三个原因是为什么压缩在本书里被深入地讨论。一些基础压缩算法提供了使信息看上去像其他一些东西的一个好方法。在开头交换算法的技术将在第6章讨论。

现在，有很多压缩数据的不同的技术正在被使用，由于这种算法的巨大经济价值，这个领域在过去的7年里已进行了很宽的扩展，一个把数据压缩一半的程序可以不用额外改变硬件而使一台计算机的存储量加倍。人们在不断地提出新的和令人惊奇的有效的压缩数据的技术，但是却又把一个能很好地完成装配数据的工作，降低为看做是一个公式的基本过程，使公式直接适合数据的参数变成了压缩的代用品。一些更流行的技术如下：

- **概率论方法**　这些方法把文件中的字符和位的出现机率加起来，然后它们把一个短的代码字分配给最普通的字符，而把一个长代码字分配给最不普通的字符。莫尔斯代码是这类中压缩算法的一个典型例子，字母“e”在英语语言中是最普遍的，作为一个点被编码。字母“p”不太普遍，所以是以一点一划相间的被编码。这个代码是这些代码中最有名的版本。
- **字典方法**　这些算法编辑了搜集最普通的字、词组或是一个文件里位的一个清单，然后给这些字编排号数。如果一个字在这个清单上，那么被压缩文件就只包含有指向字典入口的号数。如果字不在清单上，这个字就会不改变地被传送。如果数据文件有大量的文本的话，这些技术就是很有效的。一些报道声称把文本压缩原来尺寸

①例如Web TV的网络，如果要用电视机来显示，就要从图像中剥离高层的数据。

的10%～20%。Lempel-Ziv压缩算法（由以色列数学家A. Lempel和J. Ziv共同开发的压缩算法）是这种算法的最普通的使用的版本。

- **Run-Length编码** 许多简单的图像只是黑色像素和白色像素的集合块，如果沿着一条直线行走，可能会遇到1000个白色像素后面跟着42个黑色像素，再后面跟着12个白色像素，如此等等。Run-Length编码按数字的顺序进行储存：1000，42，12，...这样通常可以为黑白混合技术节省大量的空间和工作，传真机就很广泛地使用了这一技术。
- **示波图方法** 这些算法使用一个波形的集合作为导出一个公式集合的基础，然后它们把波形的尺寸和位置都调整到最佳的适合数据。这些工作非常地好，因为图像不需要被重建，只需要接近原文就可以了。JPEG、JPEG2000和MPEG图像以及视频压缩标准是这一技术中较著名的两个例子。

第14章用微波研究了信息隐藏的能力。

- **分形方法** 分形函数从很简单的公式里产生出非常复杂的模式，这意味着如果能够找到适合数据的公式的话，他们就可以获得相当高的压缩。

关于碎片压缩的介绍可以在[BS88、Bar88和Bar93]里找到。

- **自适应的压缩方法** 很多压缩方法都可以被修改以便适应不断变化的数据模式，这里描述的每一种类型是属于在数据流量的中间修改自己以适应新的模式的类型。

每一种压缩方法在特殊的领域中都是有用的，没有一个通用的算法伴随一套通用的、能很好地适应数据的函数一起发生，所以。人们就修改已经存在的算法并提出他们自己的公式。

压缩函数给那些想隐藏数据的人提供了很好的开端，因为这些函数是被建立用来描述模型的，有两种方法可以使用压缩函数成功地隐藏信息。一种隐藏数据的方法是把它铸造成其他数据的形式，这样它就混合了进去。一个在小型电子计算机上工作得很好的函数，例如，能够仿造出单色的条纹并把它们传送进一套简单的参数中。如果你有这样一个函数，它可以相反地被应用于一些数据，并且它可以把数据扩展进条纹里。结果使得数据变得更大，但是看上去会像其他的一些东西，数据可以通过再次压缩而被恢复。

压缩技术也可以用来辨识一个文件中的最不重要的隐蔽处和裂缝，以便额外的数据能够进入这些角落。很多图像压缩函数被设计成有损耗的，因此，即使重建的图像看上去可以和原来的图像非常地相似，但却不是真正的同一个图像。如果描写一个图像的函数能被更加宽松地适应，那么算法就可以使用更少的函数并且产生更小的压缩输出。例如，一个苹果可以取代一个平滑的具有多种形状的红色的连续集合体而被编码为一个红色固体。当这个图像被解压缩时，很多更小的细节就失去了，但是整个图像看上去却是好的。这些压缩函数可以很容易地把一个图像压缩到原来尺寸的五分之一到十分之一，这也是为什么它们非常流行的原因。

本章开头处的电视格式模型是一个有损耗压缩的例子。对于怎样重新建立整个程序，清单里没有足够详细的介绍。一个替代物被找到的地方，就会有一个更好的有损耗压缩的例子。

5.1.1 霍夫曼码

理解基本压缩的一个好方法就是检查一个像霍夫曼码这样的算法。这种技术是分析一个文件里的每一个字母出现的频率，然后用一个灵活长度的码字来取代它。通常，一个字母作为一个字节被储存，一个字节占8位的信息。一些标准英语平均信息量的估算，虽然显示一个字母携带3位以上的信息，但显然还有足够挤拢一个英语文本文件的八分之五的空间。诀窍就是把短的码字分配到使用普遍的字母并且把长码字分配到使用得最普通的字母，虽然一些长码字要比8位要长，最终结果将仍然更短。使用得普遍的字母将有最大的影响。

表5.1显示了美国最高法院关于字母出现机率的看法。空格键是最普通的字符，接下来是字母“E”。这个表是通过混合从低到高的大写字母来建立的。一个实际的压缩函数将为每个字母的两种形式保持相对独立的入口，就像每一种标点符号类型的一个入口一样。一般来说，每个字节有256个入口。

表5.1 美国最高法院关于字母出现机率的一组评价

字母	出现机率	字母	出现机率
空格键	26974	N	5626
A	6538	O	6261
B	1275	P	2195
C	3115	Q	113
D	2823	R	5173
E	9917	S	5784
F	1757	T	8375
G	1326	U	2360
H	3279	V	928
I	6430	W	987
J	152	X	369
K	317	Y	1104
L	3114	Z	60
M	1799		

表5.2显示了一套为使用表5.1里数据的每一个字母而建立的代码。空格键——最普通的字符，得到了一个只有两位长的代码：01。很多其他的普通字符获得4位长的代码。最不普通的字符“Z”，获得一个11位的代码：00011010001。如果这些代码被使用来给数据编码，那么应该很容易就可以把一个文件减少到比它原来尺寸的一半还少。

这里有一个简单的例子，它把在普通ASCII码（美国信息交换标准码）里用来储存一个字“ARTHUR”的48位转变到27位的压缩形式：

Letter（字母）：	A	R	T	H	U	R
ASCII：	01000001	01010010	0101000100	01001000	01010101	01010010
Compressed（压缩后）：	1000	1111	0010	00111	000100	1111

表5.2 根据表5.1建立的代码（注意：一个霍夫曼树就是基于这些代码，如图5.1所示）

字母	出现机率	字母	出现机率
空格键	01	N	1101
A	1000	O	1010
B	111011	P	000101
C	10110	Q	00011010000
D	11100	R	1111
E	0000	S	1100
F	001101	T	0010
G	111010	U	000100
H	00111	V	0001111
I	1001	W	0001110
J	0001101001	X	0001110
K	000110101	Y	0001100
L	10111	Z	00011010001
M	001100		

霍夫曼运算法则也可以用来压缩任何类型的数据，但它的有效性却是不断变化的。例如，它可能在一张每一个像素的光亮度都作为一个字节来储存的相片中被使用。这种算法在一个只有少量黑白基值的相片上是很有效的。但是，如果用很多在黑暗和名亮之间的平滑的形状使光亮度均匀地分布在一张照片上，那么这种算法就不是工作得很好。这种算法当只有很少基值的时候是工作得最好的。

还存在有很多改善的译本。创建一个第二顺序的、聚集成对的字母的代码是很普通的。这可以通过两种方法来做到。最容易的方法就是把每一对字母看成是基本原子单元，将建立一个成对表来替代建立一个字符频率表。这个表可以更大，但是它将产生更好的压缩，因为这些字母对中很多都是很少出现的，比如，“ZF”这对几乎就不存在。

另一种方法就是通过分析哪些字母跟着另外的字母来建立26个不同的表格。这样，一个字母“T”的表格就含有其他所有跟在T后的字母出现的频率。在这个表里字母“H”是非常普通的，因为“TH”在英语中经常出现。这26个表格将产生更多的压缩，因为它们可以调谐更多的代码字。字母“U”在字母“Q”后接受一个很短的码字，因为U永远都是跟在Q的后面。

这个例子显示了一个霍夫曼压缩函数是怎样在实践中工作的，它不能解释码字是怎样被建立的，也不能显示它们为什么工作得那么好。下一小节就将做出解释。

第6章显示了怎样逆向运行霍夫曼码。

5.2 建立压缩算法

创建一个新的压缩算法近来已经变成了数学和计算机科学的一个研究方向。很少有这么聪明的想法能够为人们节省数以十亿美元的存存储空间和交流时间，并且根据这些想法还

有很多更深层次的工作可做。本章将不会研究产生最好成果的工作，因为这是本书所涉及范围以外的内容，很多最简单的想法结果都最好地隐藏了信息。霍夫曼码是基本文本的一个完美的解决方法。字典算法，例如Lempel-Ziv，是成效不大的。

5.2.1 霍夫曼压缩

霍夫曼压缩很容易理解和建立。令这套字符为∑，并且$\rho(c)$表示一个特殊字符c在一个文本文件中出现的概率。通过分析资源文件来建立这样一个表格是很容易的，这通常是在case-by-case的基础上实现并且储存在压缩译本的标题中，但是它也可以预先被建立和不断重复地使用。

最基本的想法就是建立一个二进制树，这棵树的树叶上包含了所有的字符。每一个分支被标明为0或1。树根和树叶之间的通道指定了每一个字母的代码。图5.1显示了一小部分字母的树模型。

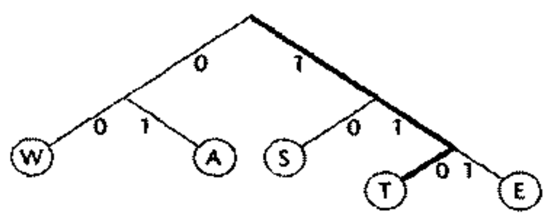


图5.1 一棵小的霍夫曼树。每一个字母的代码由树根和包含一个特殊字母的树叶之间的通道来决定。例如字母“T”获得代码110

密钥就是建立这样的一棵树，以使最普通的字母能在树的顶端附近出现。这可以通过相对容易的过程来完成，如以下所列。

1. 从每一个字符的一个节点开始，这个节点也是一棵很简单的树。这棵树的重量被设为字符联合树在文件中出现的概率。把树称为t_i，重量称为$w(n_i)$。i的数值随着树的数量变化而变化。
2. 找到最小重量的两棵树。通过建立一个新的两个分支与两棵树的根连接的节点，把两棵树粘成一棵树。一个分支将被标为1，另一个分支则将被标为0。这棵新的树的重量被设为连在一起的两棵旧的树的重量之和。
3. 重复前面的步骤直到只剩下一棵树为止。代码可以通过跟随树根与树叶之间的通道来建立。

我知道一个希腊式的迷宫，它是一条单独的直线。其实一个简单的测试就可以走出迷宫，但是很多哲学家沿着这条直线都迷路了。

——Jorge Luis Borges, *Death and the Compass*

首先，最小重量的字符被连接在一起了，每一个连接过程给树根和树叶之间加上了另一个层，这样就很容易看到最不普通的字母是怎样被推到远离树根的地方，而在树根处这些字母有一个很长的码。最普通的字母直到最后才会合为一体，所以它们都是在顶端出现的。

这种算法总是把最小的重量排在第一位，所以它很自然地保持了树的平衡。一棵树的重量代表任何树中的字符在一个文件里出现的次数。设想一下，如果在某一步骤中错误地选择了两棵树连接在一起后会发生什么情况，就可以证明用这种算法建立的树是最好的。越普通的字符就被推到离树根越远的地方，并且比不普通的字符得到更长的代码，平均的压缩下降了。图5.2是基于表5.2中的代码的一棵霍夫曼树。

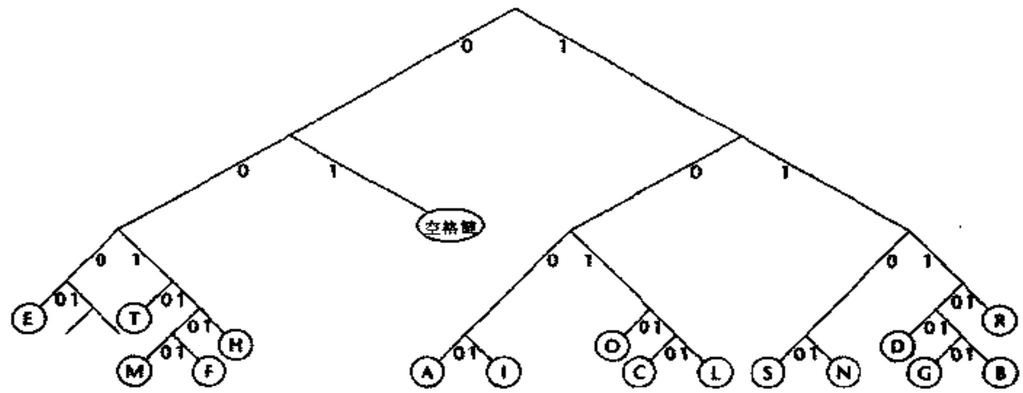


图5.2　树的顶端是根据表5.1的数据建立的，这样产生了表5.2显示的代码。考虑到空间问题这里只有顶端部分显示出来。像"Z"这样的最不普通的字母在这棵树的相应部分用虚线来代替

还有很多其他的人通过创建一些使用一棵树的节点地址的算法，从而扩展了霍夫曼编码的主题。一种流行的技术就是使用"扩张树"。这些树在每一个字符每被编码一次就被修改一次。一个版本在一个包含有很多结构的复杂移动中把字母移到了树的顶端，结果最普通的字母就像冒泡一样跑到顶端。树的连续重复排列意味着这棵树适应了当地情况。这种类型的算法对压缩一本字典是理想的，这本字典在章节中即使是像"j"或"z"这样的最不常用的字母也是普通的。当这种算法移动到字典的"j"部分时，包含"j"的节点就将被重复地推到"扩张树"的顶端，那里它将获得一个短的代码。随后，当压缩函数到达了"z"节，"z"节点就将终止于树的顶端附近，照例也给"z"一个短的代码。很显然，这种压缩方法的一个主要问题就是整个文件的压缩过程从一开始就必须保持一个"扩张树"的精确的说明，你不可能简单地跳到"z"节点并开始解压缩。

[Sto88]是一个关于压缩的很好的基本参考书。

霍夫曼基本算法也有很多不同的用处。在第6章中它被用来把数据转变为看起来像英语文本的东西，在第7章中它也被使用来在不同的字之间做出任意的选择。那里和这里一样，同样的结构都是有用的。

5.2.2 字典压缩

像Lempel-Ziv算法这样的流行和具有潜力的压缩方法被称为"字典方法"，因为它们在一个文件里建立了一个挺大的普通字的清单①，这个清单可以在压缩一开始立刻建立，也可

①这个算法对压缩文件并不是特别好，就像霍夫曼算法用于字典一样。事实上，我使用的是一个与前面一节的例子相同的规则的字典，这只是一个巧合。不要因此而困惑。

以随着算法在文件中执行过程慢慢适应地建立和改变。这种算法是成功的，因为一个在字典中描述位置的指针比普通字本身占用了少得多的空间。

字典就是一个字的清单。它几乎总是有2^n个字，因为每一个特殊字的指针都占有了n位。每一个字既可以是一个固定的长度也可以是一个灵活的长度，长度固定的是很容易处理的，但是灵活长度的在估算英语语言和x86机器代码上则做得更好。

压缩是很简单的。首先，文件被分析以建立一个有2^n个最普通的字的清单，然后对文件进行从开始到结束的扫描。如果当前的字是在字典中的，那么就在字典中紧跟着字的位置用一个标签<InDict>来代替。如果当前的字不在字典中，那么就在紧跟着保持不变的字的后面用一个标签<Verbatim>来代替。

显然，这种算法的成效，依靠于标签<InDict>和<Verbatim>的大小、字典的大小以及在字典中找到的那些东西的出现次数。一个简单、通常、有效的解决方法是使标签占有整个字节B，如果字节的数值为0，那么下n个位在字典中就代表一个字；如字节B的数值比0大，那么原文件中就会逐字地复制出B字节。这种方法允许程序使用在英语上工作得很好的灵活字尺寸。在一些情况下有很多更有效的不同的方法。

字典的索引并不需要n位数目，也可以计算字典中字的出现机率和使用一个像霍夫曼一样的方法来给它们中的一些设计短的代码。在这种情况下，逐字的文件标签通常包括另一个字。

字典也可能在文件执行过程中适应。一个简单的技术就是保持字典中使用的最后一次的那个字的轨迹，无论什么时候，遇到逐字的文本中的一节，最旧的字就会被换出字典而最新的逐字文本就会换入字典。这种技术对文本来说是很适用的，因为很多字都在章节中群集分组。例如，字“dictionary”，“Huffman”和“compression”在此节中是很普通的，但是在本书的其他部分却是相对比较罕见的。当这些字被第一个遇到并且直到有一段时间没有被使用的时候，一种适应的方法就会把它们加载进字典中开头的章节。

字典方法对压缩任意的文本是很有效的，但是它们很难相反地运行以使数据能模仿一些东西。第6章使用像霍夫曼一样的算法产生真正的文本，但是它不包括有关反转字典算法的那节。它们是在本章中被描述的，因为压缩是节省空间和使数据变得更白的一个好方法。这种算法对模仿来说并不是工作得很好，因为它们需要一个构造地很好的字典。在实践中，没有很好的、自动的方法——至少就我所知——能建立一个好的字典。

5.2.3 JPEG压缩

在5.2.1节里描述的霍夫曼编码和在5.2.2节里描述的字典方法对任意数据的搜集是很理想的。它们在某些图像类型文件中也工作得很好，但是在其他方面却很失败。如果一个图像在一个可预测的模式里出现很少数量的颜色，那么这两种算法都能够很好地完成找到一个良好压缩模式的工作。当图像包含有很多不同的颜色形状的时候，这就不会发生。例如，日本的国旗，有一个连续的颜色，就是一个红色的圆，而苹果实际上则有很多不同的红色形状。

在怎样使一个算法协调一个特殊的数据类型方面，JPEG算法是一个很好的例子。这种方法把余弦函数装配到数据上，然后储存余弦函数的振幅和周期，使用的函数的数字和尺寸可以根据所想得到的压缩的数量而不断变化。除了木纹状的的图像外，少数的函数可以产生

很大数量的压缩。更多的函数可以增加准确性，但是占用了更多的空间。这种灵活性是有可能的，因为人们并不是特别在意当图像进行解压缩的时候，他们是否能得回同样的、“正确”的图像。如果它看上去足够接近原图像，它就算是够好的了。

这种灵活性对于JPEG编码是多么的有用！当5.2节里的算法能够反向运行并产生模仿英语文本的时候，JPEG算法就不能工作得很好。尽管如此，它还确实有辨识那些具有保存信息的空间的图像里的隐蔽处和“裂缝”的能力。这在第9章中将被详细地描述。

5.2.4 关于GZSteg

很多压缩算法可以用聪明的方法来调整以隐藏信息。其中一个最简单的，但不是最有效的技术被Andrew Brown使用了，那时他提出了流行的GZIP压缩算法。这一技术在网络上不断地被使用，所以它成了一个无害位置的理想选择。

通常，GIZP算法是通过在指向数据被找到的先前的位置插入记号来压缩数据的。这里有一个文本章节的样品：

> The famous Baltimore Oriole, Cal Ripken Jr., is the son of Cal Ripken Sr. who coached for the Orioles in the past.

下面这里有一个被压缩后的章节样本。用斜体来做标记。

> The famous Baltimore Oriole, Cal Ripken Jr., is the son of *(30,10)* Sr. who coached for the *(48,6)*s in the past.

在这个例子里有两个标记。第一个是（30,10），它告诉算法往回走30个字符并且复制10个字符到当前位置。压缩技术对很多文本算法都工作得很好。

GZSteg通过对复制件改变字符的数目来隐藏信息。每插入一个标记就需要超过5个字符被复制，它将隐藏一位。如果位是0，那么标记就保持不变。如果位是1，那么被复制的字符数目就会一个一个变短。这里用两个位11被编码作同样的引用：

> The famous Baltimore Oriole, Cal Ripken Jr., is the son of *(30,9)*n Sr. who coached for the *(46, 5)*es in the past.

在这两个情况中，复制的尺寸被切成了一半。这样就把压缩的数量减小到了一个很小的程度。

这种方法的最大优点就是文件格式不会被改变。一个标准GZIP程序不用在解压缩过程中揭示所隐藏的信息就可以解压缩数据，信息可以不引起怀疑而被留下。尽管如此，一个很快的分析也可以揭示出在这样一个方式里隐藏的数据。如果扫描这个文件并检查标记，就可以很容易地决定哪个标记是最小的字符。在GZIP压缩有一个错误时，没有办法能够否认这个程序。

5.3 小结

压缩算法通常被用来在不删除信息的情况下减小一个文件的尺寸，这可以增加它们的平均信息量和使得文件显得更加随机，因为所有可能的字节都变得更普通了。压缩算法当被用来产生仿真时也是很有用的，这还会在第6章描述。

- **伪装** 压缩算法一般产生看上去更随机的数据。也就是说，会有更好的数据分配。
- **安全性** 它并不是完全安全的。大多数压缩算法在文件的开头就传送表格和字典，这可能不是必需的，因为双方都在事先同意了这样的一个表格。虽然不知道怎样算出在霍夫曼算法里字母和位之间的映像，但相信这是不难算出的。
- **怎样使用** 对所有计算机来说，有很多压缩程序可以利用。通常使用比这里提供的版本更好、更合适的算法，它们可以为任何加密程序找到一个理想的首次通行证。

第6章 基础模仿

在诗行之间阅读

以下是一张报纸的人物专栏中一个愤世嫉俗者的读后感的抄本：

> wants same for walks in rain, trips to Spain and riding on planes. Send picture and dating strategy.

（这段话中文的意思是：要想如同在雨中漫步，坐飞机去西班牙旅行。寄张相片并注明约会的方法。）

伟大。Eliza Doolittle。照字面意义。我提出了一个约会的方法，而她除了沿途乘车之外什么也不做。这不是一个好迹象。她大概是一个像秘书一样工作的公主，想被模教并送入富裕的高层阶级中。如果判决我出局，我将去干我已丢掉的大部分工作，这样她在西班牙就可以更轻松地尝试不同的单词发音。不管怎样，西班牙有多么浪漫啊？她也可能再读海明威的作品，而我将被迫在一群公牛前面奔跑，以致于她认为我是在作旧式的冲撞。不用谢。我将采用一种新式的冲撞，就像“Range Rover”（山地漫步者，一种车的牌子）奔驰在非洲一样。不会被一群公牛撞倒。并且，如果是下雨的话，我将不会被淋湿，也不会在任何地方摔倒。

> SF ISO SM. Dancing. Wine. Night. Sky. Moon. Romancing. Dine. Write by June.

（中文的意思是：跳舞。葡萄酒。夜晚。天空。月亮。浪漫史。用餐。由June写作。）

这真是伟大的诗。她期望我的报答，但我不能说：“好，让我们今晚抢夺一个汉堡包。”不是的，我将不得不给她一封用水彩描绘的信。在一些古老的诗文形式里，压韵是一种过度活跃的思想的标志。如果一个段落的不同部分的两个字碰巧是以同样的音结束，那谁会在意呢？这只不过是一种巧合而已。她可能会花所有的时间来改进、增加我们生活中的格调，我却不得不一直这样说：“不。我仍然很爱你。我只是想看世界职业棒球大赛的第17场比赛而已。今年“红短袜”队也在里面，他们可能实际上赢了！这并不是一个关系糟糕的迹象。”

> and fast horses are for me. Don't write. Send a telegram.

（中文的意思是：快马是给我的。不要写信，发一个电报。）

快马？它们是动物，只有当燕麦是新鲜的，它们才容忍我们骑在它们的背上。女人也一样，但不能用缰绳控制她们，而且她们不仅想要新鲜的燕麦。我打赌快餐一定不在她的食物清单中。她将骑马向前并抓着我，因为我哪儿都去过。于是，她将一个快速航班从我的生活中夺走。她的小船已经在下沉。

6.1 反向运行

寻找本章中所描述的一个约会，努力做一个简单的广告，并在诗行之间跟随它的规则，阅读出它画出的全部的关系弧线。个人广告有一个能把一个人的梦想压缩到100个字以内的精巧的简写系统。这些年里这种简写是随着人们所想要的模式不断地改进来发展的。"ISO"意思是"In Search Of"，如此等等。愤世嫉俗者只是在使用他的看法：人们想把数据位扩展进一个现实的与输入数据无关的实体中。

本章的主要内容是提供一个自动的、很小的、无害的数据位的方法，以及用深藏的、经过修饰的细节来布置它们，直到最终能完全不同地模仿了一些东西，数据就像如同穿上服装一样被隐藏了。在本章里，处理的结果是通过对第5章里描述的霍夫曼压缩算法反其道而行之完成的。通常，霍夫曼算法和文本的统计分配近似，然后把它转换为一个数字简写。反向运行霍夫曼压缩算法可以使普通数据形成这些精巧的模式。

图6.1是一个很好的开端。这个图的文本是通过分析第5章的早期草稿而使用第5顺序规则模仿函数来产生的。本章中的第5顺序统计规则，是通过计算文中所出现的每行中所有可能的5个字母来产生的。在这个草稿里，5个字母"mpres"在那个顺序中出现了84次，假设这些字母是字"compression"的部分，那么5个字母"ompre"和"press"也出现84次是并不惊讶的。

这个文本是在一个由这些统计值指导的过程中产生的，计算机是通过随机选择5个字母一组来开始的。在这个图表里，第一组5个字母是"The l"（空格算一个字母）。然后它使用统计值来指示随后的字母。在第5章的草稿里，5个字母"he la"出现了两次，字母"he le"出现16次，而字母"he lo"出现了两次。如果第5顺序文本是要仿真第5章的统计轮廓，那么字母"a"跟在"The l"后面的机会在20次中就有两次。当然，"e"的机会应该是20次中有16次，"o"的机会20次中应该有两次。

这个过程被无限地重复，直到有足够的文本产生。结果真实度通常令人很惊异。在很大程度上，这是由更小尺寸的样本文本引起的。如果假定在一个文本文件里有64个可印刷的字符，那么就有645种5个字母的不同组合。很显然，它们中有很多像"zqTuV"一样从不在英语语言中出现的组合，但是如果算法有很多个选择的话，它们就有很多必须进入表格的路径。在最后一个例子里，跟随"The l"的一个字母有三种可能的选择，短语"The letter"在第5章中是很普通的，但是短语"The listerine"就不是。在很多情况下，在样本里使用的小数量的字只指定了一种可能的选择，即给它一个这样的真实探通术模式。

这里是产生由一个源文本*S*提供的名为*T*的第*n*顺序文本的一种算法。

1. 建立一个清单，在这个清单里有在*S*中出现并保持*S*中每一次出现的轨迹的所有不同的*n*个字母的组合。
2. 随机选择一个组合作为种子，这将是第一组*n*个字母的组合*T*。
3. 重复这一循环直到足够的文本产生：
   (a) 选择最后$n-1$个字母组合*T*。
   (b) 搜寻统计表并找到所有以这些$n-1$个字母开始的字母组合。
   (c) 这些组合的最后字母形成了一套为加在*T*上的下一个字母的可能的选择。

(d) 在这些字母中进行选择并且使用它们在S中出现的频率来衡量你的选择。

(e) 把它加到T上。

```
The letter compression or video is only to
generate a verbatim> followed by 12 whiter 'H'
wouldn't design a perfective reconomic data. This
to simple hardware. These worked with encodes of
the data list of the diction in the most come down
in depth in a file decome down in adds about of
character first.
Many data for each of find the occnly difficular
techniques can algorithms computer used data
verbatim out means that describes themselves in a
part ideas of reduce extremely occurate the charge
formulas. At leaf space and the original set of
the storage common word memo to red by 42 black
pixels formula that pression of their data is why
complicated to be done many difference like
solution. This book. Many different wouldn't get
into any different to though to anyone has make
the popular to the number or 60 minutes. This
Huffman also just but random. Compression. One
branches is easy to be use of find the because
many people has shows the codes The most nooks
like three constructed with a function, the greate
the moMany good formations. This simply be
compression show a Huffman code work quite easily
common in these 26 different takes 48 bit should
in this can be patter-frequency the image space
constructed in the other letter is algorithm on
stand there easier to the overed into the root and
MPEG and crannies their data for compression
Scheme Compression in a file description when it
short codes were could be common length encode
work quite weights a Klepto Family Stressed by
image and Compressed as a bigger. whiter the for
hardware. Many even more that then the result to
descriptionary algorithms that were two bits you
might for simply because of charge found in the
well, but the data is easily Stressed surprising
text. The algorithm would look very good
```

图6.1 这是一个第5顺序随机文本，是通过模仿第5章的一个早期的草稿里统计分配字母而产生的

这种算法是在n等于2或者比2大的条件下工作的。显然，更低顺序的样本输出质量依赖于顺序。这里有一些样本：

- **第1顺序**：islhne[hry saeeooisnre uo ' w nala al coehhs pebl e to agboean ce ed cshcenapch nt sibPah ea m n p [tmsteoia lahid egnndl y et r yf arleo awe l eo rttnntnnhtohwiseoa a dri 6oc7teit2t lenefe clktoi l nlte r ces. woeiL, isetemd2np eap haled&oolrcc yttr tr,oh en mi elarlbeo tyNunt. syfes2 nrrpmdo,0 reet dadwn;dysg te.ewnlca-ht eitxrni nttos xt eCc oh sao vhoh0mhgr
- **第2顺序**：Thy etheren' ante este ales. icone thers the ase omsictorm s iom. wactere cut le cc s mo be t Me. Y whes ine odofuion os thore cctherg om tt s d Thm & tthamben tin'ssthe, co westitit odecra fugon tucod. liny Eangem o wen il ea bionBulivethe ton othanstoct itaple
- **第3顺序**：ith eas a tan't genstructin ing butionsmage ruct secate expachat thap-res "Miamproxis is of is a to af st. This there is monst cone usectuabloodes it a luengettecte por be the andtaly com Bevers gor the Hufferess. M B G achasion the coduch occomprence mon Quited the ch like bitheres. The
- **第4顺序**：captionary. Image and to compression lest constance tree. Family for into be mode of bytes in a lgorith a file of that cosition a lgorithm that word even that a size smmarge factal size are:

  ite postion scien Raps.

  The is are up much length ence, the if the a refsec-ent sec-ent of fits to the crans usuall numberse compression

  A good ways that in a lgoright. The brase two wants to hidea of English Cash the are compres then matimes formatimes from the data finding pairst. This only be ression o

毫无疑问，随着顺序的增加，文本会得到越来越多的可读性。但是谁将是受骗者呢？如果敌人设计一个计算机程序，它会通过辨识不合规律的字符混合来给可疑的电子邮件做标记，那么又会怎样呢？外国语言在这方面可以很突出。例如法语，和不同的字母分类一样，有很多省字号。俄语则有一个完全不同的字母表。每一种语言甚至每一种地方方言都有一个不同的成分。

这里产生的这些文本可以欺骗这样一个扫描装置，其输出与诚实的英语文本在统计上是相等价的。例如，字母“e”是最普通的，“t”是下一个最普通的字母。在所有不同的顺序里每一件事看上去在统计上都是正确的。如果扫描软件正在寻找统计偏差，那么它肯定找不到结果。

一个自动的扫描程序在与相对短的文本样品比较中也处于不利的统计地位。它关于什么是标准的统计的定义，必须足够松散到能适应由文本焦点引起的变化。例如，一个有关zebra的文件将有比普通文件更多的字母“z”，但是这并没有使它变得不正常。有些文件“j”和“q”的出现机率通常比平均值高，这是因为主题包括了一些像监禁（英文jails）或智力竞赛（英文quiz）节目的东西。

当然，这些文本还不能欺骗一个人。至少第1、第2或第3顺序文本不能欺骗任何人。但是一个像法律文书这样的有着许多模糊的、难懂的术语的一个第5顺序文本，却可以欺骗很多对此类文本结构不是很熟悉的人。

更复杂的语法分析当然也是可能的，有语法检验器能够扫描文件并辨识出坏的句子结构，但这些产品距离完美还相差甚远。很多人使用惯用语句来写作，还有的人扩展了自认为不会破坏任何规则的语法范围。虽然人们产生的诚实文本可以设置很多标记，毫无疑问甚至是图6.1所示的第5顺序文本也会出现错误，结果是错误会被自动地察觉。可以这样说，任何有比正确文本更多错误的文本，都会通过一个自动过程被标记为可疑的[**KO84，Way85**]。

第7章提供了一个使语法检验器失效的方法。

6.1.1 选择下一个字母

本章的最后一节展示了怎样通过仿真一个源文本集合的统计分配，从而产生与正常规律统计等价的文本。运算法则展示了怎样选择下一个字母以便它在规律统计上是正确的，但是它却没有解释怎样在过程中隐藏信息，也没有解释怎样反向运行霍夫曼压缩。

信息的隐藏是通过被隐藏的数据指示如何选择下一个字母而实现的。在上面描述的例子里，“a”、“e”或“o”可以跟着开始字母“The 1”的后面，这样就很容易提出一个对信息进行编码的简单的方法。如果“a”代表“1”，“e”代表“2”，“o”代表“3”，那么可以通过字母的选择对普通数字进行编码。如果相隔一定距离的某些人有一个产生统计表格的同样的源文本S的副本，他们就可以恢复这个数值。他们可以查找“The 1”并发现在表格中有三个字母跟着“he 1”，字母“e”是依字母顺序的第二选择，所以字母“e”就代表信息“2”。

像图6.1所示的一个长的文本可以在每一个字母中隐藏一个不同的数字。如果被加到输出端的下一个字母没有选择，那么就没有信息会被隐藏，那个字母就不会隐藏任何东西。

简单地使用一个字母给一个数字编码并不是发送数据的一个有效的和灵活的方法。如果想发送信息“3”而只有三个选择的话，怎么办呢？如你的数据想要发送数值“1”，但是第一个字母却是最不普通的选择，又怎么办呢？这会搞乱统计成分吗？

反向运行霍夫曼编码是解决上述所有问题的方法。图6.2展示了一棵由跟在“The 1”后的三个字母选择建立的简单的霍夫曼树。图中显示了字母“e” 的被跟次数在20次中有16次，而字母“a”和“o”被跟次数都是两次。使用这个统计规律，我们就建立了这课树。

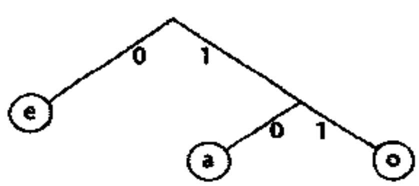


图6.2 一个小树，它被建立用来隐藏一个新的字母选择的位。其中字母“a”编码为“10”，字母“e”编码为“0”，字母“o”编码为“11”

信息就是使用像这样的一棵树用很多不同的位进行编码的。“e”的选择编码为位“0”，“a”的选择编码为“10”，“o”的选择编码为信息“11”。这些位可以通过反向选择在另一端（接收端）被恢复。用一个字母的每一种选择隐藏的位数目，比直接地用可能的和统计选择的概率的选择数目更加多样化。

一般来说，如果源文本S足够大，大到能提供一些偏差，那么就应该有多于三种选择可以被利用，但是很少有达到26种选择。这是很自然的，因为英语对语言的构造有充足的冗余。Shannon在他提出信息理论的时候就承认了这点。如果英语的平均信息量大概是每字符3位，那么这意味着至少有2^3或8种能够制造下一个字符的选择。这个数值是通过概率论来衡量的。

当然，这个方法也有一些问题，仍然不是隐藏信息的最好的方法。它是因为同样的原因而仿真源文本S的，这个原因是霍夫曼编码是建立像树一样的压缩方法中最有效途径。同样的证据从反面也显示了这个操作。

6.2.1节“利用额外的数据”展示了一个更精确的近似值。

如果反向运行霍夫曼编码是最好的解决方法的话，就没有完美可言了。在图6.2中的那个例子里，如果下一个将被隐藏的位是“0”，那么字母“e”就被选择。如果下一个位是“1”，那么字母“a”或“o”将被隐藏。如果将被隐藏的数据是纯粹随机的，那么“e”被选择的次数将有50%，而“a”或“o”被选择的次数就是另外的50%，这不会是真正地从源文本里仿真出来的统计值。如果真是这样，字母“e”被选择的次数就有80%，而其他每个字母被选择的次数只各为10%。鉴于霍夫曼树的二进制结构和可利用的选择数目，所以说这个不准确度是存在的。

6.2 仿真的执行

在编写能产生合格的第n顺序仿真的软件时，会有几个主要的问题。首先是获得并储存统计值。第二是产生一个树结构以完成像霍夫曼一样的编码和解码。第一个问题需要更多一些的技巧，因为有几种不同的方法来完成同样的结果。第二个问题则是相当地简单。

几个不同的人已经同样地提出了被称为产生一个模仿的问题。这可以在描述怎样产生统计等价的文本的一系列杂志文章[KO84，Way85]中找到，这些文章并不使用这个结果来隐藏数据，但是它们集中最有效的方法来产生它。在实践中，这个工作的结束与同音密码相似，这个同音密码是由H. N. Jendal，Y. J. B. Kuhn和J.L. Massey在[JKM90]中描述的，并且是由C. G. Gunther在[GUN88]中归纳的。

这里是几个储存需要产生数据的统计表格的不同的方法：

- **巨大的排列数组** 用c^n个盒子来分配一个数组，这些盒子里c是在每一个位置可能出现字符的数目，n是正在被保持的统计顺序。显然，如果只有大写字母和空格被保持，c就可以低到27。但是，如果一个字节的所有数值都被储存，那么c也可能是256。这对小数值n来说可能是很实用的，但它不可能很快地增长。
- **巨大的清单** 产生所有条目的一个依字母顺序的清单。和一个指示器一样，每一个节点都有一个计数器，使一个存有数值的字符串被审查。这实质上使得节点比数组更无效。如果有很多节点被排除在外，这也可以成功。如果英语文本正在被仿真，就有很多几个字母的组合不会出现。一个清单可以确切地使操作更加有效。
- **巨大的树** 建立包含一条从树根到树叶的路径的一个大树，以便每一个字母组合在树中都可以找到。这实质上可能包含更多的指示器，但是，使用它比使用巨大的清

单更快。图6.3阐述了这样一个运作机制。

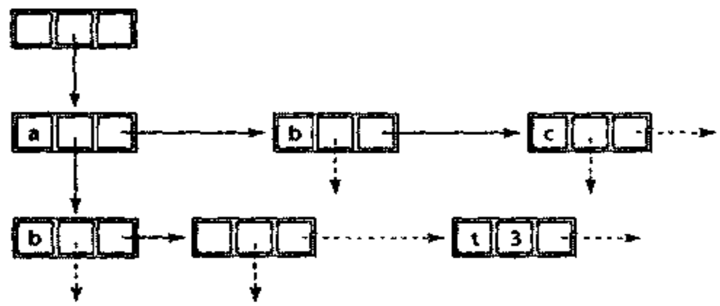


图6.3 这课树储存了一个文件的数据频率，这个文件是第n顺序统计值的具有n层分支的文件。访问实质上是更快了。虚线显示的是节点被遗漏的地方。这里所示的惟一一个完整的字是"at"，它在样本中出现了3次

- **钓鱼** 像鱼上钩一样，使搜索随机化，没有统计表格产生，因为c和n都太大了。源文件是作为一个随机源而服务的，并且它为每一个新的字母的选择做随机参考。这可能会相当慢，但是，如果存储器不能利用的话，它可能是惟一的选择了。

在上述解决方法中，前3个对任何具有标准程序背景的人来说都是很容易实现的。数组是最容易的，清单也不难。任何人都有很多选择来完成这棵树。图6.3显示了在一个清单里储存的每一个层次的新的分支。这也可以在一棵二进制树中做到并加快查找速度。

第4种解决方法"钓鱼"则有一点更加复杂。这一想法是在文本里随机选择位置并使它搜索随机化。由于所有的数据都不能在一个表格中被保存，所以，所有的选择在每一个连接点都不能被利用。因此，必须使用能够找到的东西。这个算法的最极端的版本能搜索整个文件，并很快地建立正确的表格入口。这是一个更切合实际的方法：

1. 在一个源文件中随机选择一个位置。把它称为字符i，这个随机源必须在解码过程中被复制，所以它必须来自于一个同步的伪随机数字生成器。
2. 如果正在建造一个第n顺序仿真，向前搜索直到找到需要注意的$n-1$个字符，下一个字符可能就是你想要的那个。
3. 我们让源文件中有k个字符，去决定$i + k/2 \bmod k$的位置，向前搜索，直到$n-1$个正确的字符被找到。
4. 如果两个位置暗示的下一个字符是相同的，那么这个位置就没有什么可以被编码。发送那个字符并且重复做。
5. 如果它们是不同的，那么一个位就可以通过这个选择来编码。如果你使用这种仿真来隐藏一个0，那么被找到的输出字符则是以位置i为开始的。如果正在隐藏一个1，那么搜索后找到的输出字符就是在位置$i+k/2 \bmod k$开始的。

这种解决方法可以被解码。只要编码器和解码器都能访问同样的源文件并且同样的i数值通量来自于同一个伪随机源，那么，这里所有被编码的信息就都可以被恢复。伪随机生成器保证了所有可能的结合是不可掩蔽的。尽管这样，还是假定候选的$n-1$个字符能分配在整个文本里。

这个解决方法也可以扩展到每一个输出的字母储存量多于1位。你可以在4个不同位置开始搜索，并希望能在输出端把4个可能的字母揭示出来。如果你这样做了，那么你就可以编码两个位。这个方法可以延伸到更远，但是每一次搜索都使输出变慢了。

一般来说，钓鱼的解决方法是所有方法中最慢和最麻烦的。查找每一个新的字母花去了大量的时间，这与数据中$n-1$个字符组的出现是成比例的。数组方法的查找是最快的，但是在很多情况下它禁止查找大的数。树的方法查找起来是第二快的，并且一般对文本应用可能是合乎需要的。

6.2.1 利用额外的数据

令人沮丧的是，统计的纯度通常很难产生。如果被隐藏的数据有最大平均信息量（译者：或称熵），那么从以霍夫曼树为基础的仿真中出现的字母，将呈现出一个好像有点可疑的大概分配。每一个字母将以1/2i的形式概率而出现，或者占50%、25%、12.5%，等等。这可能不是很重要，但是它可以被察觉的。

通过交替使用一些有效的位，以及使用一个伪随机数据生成器增加更多的位，可使我们的选择与数据的正确出现率更加相近，从而可以得到更好的结果。

这个技术可以用例子作最好的解释：设想有三个字符，“a”，“b”和“c”，它们的出现概率分别为50%、37.5%和12.5%。普通的霍夫曼树看上去像图6.4所示的那样，字符“a”将在输出文件里出现的次数为50%，这是很不错的。但是“b”和“c”都将出现25%的次数，字母“b”出现的频率和“c”的一样，并不是源文件中所示的是“c”的三倍。

音乐也是一个公平的比赛。很多音乐都有过使用音乐规则成分，从已存在的音乐静态模式中产生新的音乐的经历。[BJNW57]是有关这个主题的一本书。

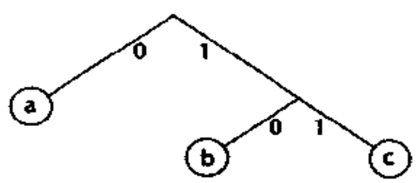


图6.4 一棵为三个字符“a”，“b”，和“c”而建立的普通的霍夫曼树，这三个字符出现的概率分别为50%、37.5%和12.5%

而图6.5则展示了一个设计以平衡分配的新的霍夫曼树的样本。现在有两个额外的层被加在树上，分支选择使得这两个额外的层要使用由一个伪随机数据生成器提供的额外位。当它们被恢复的时候，这些位就会被丢弃。如果正在隐藏的数据被很好地分配，那么建立使得“b”和“c”将出现37.5%和12.5的次数应该是很容易的。

这个过程的花费是很有效的。新的树可以产生正确分配的输出，但是解码通常是不可能的。字母“b”从地址为100、101和110的树叶中产生。既然在图6.5的树中只有第一位是保持连续的，那么只有一个位可以用字母“b”来隐藏。另外两个位可以由伪随机位通量产生并且不能在另一个末端被恢复。图6.4的树可以用字母“b”隐藏两个位，但是它将产生一个25%的机率。这是比精确度有效的交替互换。

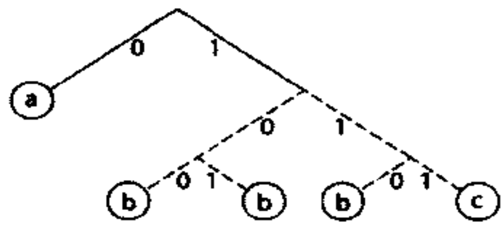


图6.5 树的一个扩展的样本。通过由一个额外伪随机数据源选出的位来决定哪一个虚线分支将被使用，欲使用根据将被隐藏的数据而得来的一个位，则只有第一种决定可以选择

如果输出为一个“c”，那么有多少位被隐藏或编码呢？当一个111在输出文件里被找到时，也可能是3被编码。或者当输出为字母“b”时，以同样的方式可能只有一个位填补。这两种方法的任意一种都是很好的。

这个技术很有意义，能被扩展到支持任何数量的精密度。最重要的步骤就是要确保在解码过程中没有模糊之处。如果在两个分支中存在同样的字符，那么使用任何从这个点下来的子树都没有位能被编码。

这意味着由一个出现次数多于50%的字符支配编码数据是可能的。如果“a”、“b”，和“c”的出现机率各自为75%、25%和5%，那么使用这种方法编码信息和产生出现机率为75%的字母“a”是不可能的。

根据这个过程，我们认为有一种方法是产生成对的字符。如果是一个字母支配分配的话，这通常是可行的。那就是说，可以用概率分别为56%、18%、3%、18%、6%、1%、3%、1%和2%来产生成对的“aa”、“ab”、“ac”、“ba”、“bb”、“bc”、“ca”、“cb”和“cc”。

6.2.2 规则仿真和图像

本章中阐述的规则仿真算法是针对文本的，并且它们在这个领域里做了一项很好的工作。只要被数字化的图像是两个字母“1”和“0”的模式，那么把它们采用到图像中就是有可能的。但是，这种成功在某种意义上来说意义不大。

第9章展示了怎样翻转最小有效位以储存信息，该章并没有试图仿真最小有效位的模式，它只是假设属于一个标准的偶数分配。规则仿真算法可以被用来设计一些模型的分配。

最简单的解决方法是把像素集合在一起形成一个规则数组。这些数组可能是2×2或3×3块，或者它们可能被破坏成从同样的行里而来的直线像素数组。现在，这些像素的每一个最小有效位能作为字符来对待，一个图像可以被用来作为计算产生一棵霍夫曼树的分配表格的模型，然后使用这棵树产生取代图像中的最小有效位的位块，这样数据就以另一种方式被隐藏了。

更具改进意义的解决方法可以以像素本身的颜色为基础，但是这个解决方法在实践当中太复杂了。这个系统的优点是当一个图像被产生的时候，它能察觉并模仿任何被不规则引入的统计值。通常，CCD（Charge Coupled Device，电荷耦合装置）阵列如有一点不完整性，它们对每一个感觉细胞对光的反映就会有相当的影响。像NASA（National Aeronautics

and Space Administration，美国国家航空航天局）这样的组织使用的高质量的天线阵是被测试和校正过的，而大多数民用的天线阵从来不进行处理。如果它们碰巧集合在反映像素组的统计分配里，这个系统就可以改善任何低层次的不适应性。

6.3 小结

本章描述了怎样产生看上去与原文本在字母统计方面相似的仿真文本，本章里描述的机制把字母看成独立的元素，一些东西允许数据通过一些统计测试但是却通过不了另外一些测试。像“e”和“t”这样的字母的统计数可以是一致的，但是通常在字典中有很多字都不能被查找到。通过一些实验而采取的另一种方法把字作为单独的统计模式的元素来对待，这需要更多的文本来产生模型，如果它是散漫的、不连贯的话，它就能提供极好的结果，没有拼错的字不能在源中被找到。

第7章描述了怎样使用一个更改进的、基于语法的方法来获得一个更好的结果，第8章则讨论得更加深远，并且展示了一个图灵机（一种装置的某种数学模型，它能够改变内部状态，并按照当前的状态在一个潜力无限的磁带上进行读、写和传送，从而构造出类似计算机特性的模型）怎样向前和向后运行以产生最复杂的文本。

- **伪装**　由这些规则仿真函数产生的文本可能是非常逼真的，其结果相对于统计是等价的。第一顺序文本将与第一顺序统计值相似，第二顺序文本也将有同样的出现机率。在更高的顺序里这还更加逼真，但是它很少能通过阅读测试，霍夫曼码将很快认出它并把它作为杂乱信息。
- **安全性**　没有人去猜测这个系统比信息隐藏提供了任何更多的安全措施，破坏这样一个统计系统的难度问题是一个公开的问题，相信检查统计值并提出一个关于用来产生文本的霍夫曼树的形状的较好猜想是有可能的。如果一些已知的简单文本可以被利用的，那么可能有上千个选项可以被很快地测出。因此，这个系统应该可以在要求逼真但并不完美的低级应用里使用。
- **怎样使用**　现在没有软件能正确地处理这个问题，但是它应该是很容易编码的。

第7章 语法和仿真

每一天发生的事情

最近，我和Charles Radw坐下来聊了一会儿，他是一个进化论科学家。我问他对进化怎样影响我们生活的其他方面有什么看法。以下是我们的交谈内容。

问：所有的洗手间都需要靠手动装置来使水停止流动。这是为什么？

答：发明这种设施很显然是为了阻止钙化作用。额外流动的水会使蓄水池中断，并且具有这个因素的洗手间比那些没有这个因素的洗手间会更经久耐用。这是简单的自然选择。

问：那么烤面包机又如何呢？不管你多努力地把它们设定到正确的状态，它们总是会把面包烤焦。

答：烤面包机做出这种反应是为了保护它们的主要的有机体。金黄色的面包需要得到一个很浓的黄油覆盖，漂亮的烤面包机给它们用户带来心肌梗死，这就使它们变成一堆垃圾。最好的烤面包机是那些不会对用户有害的面包机，它们将占据厨房中一道美丽的生态风景线。

问：我开灯的时候电灯泡总是会烧掉。为什么它不会在发生异常的时候才烧掉呢？

答：电灯泡在一开灯时烧掉，这也是为了保护它们的用户。人们进入一个黑暗的房间的时候会开灯。如果电灯泡在这个时候烧掉，那么就没有人会陷入黑暗的困境中。但是在某个人已经在房间的中央的时候电灯泡才烧掉，那么那个人就总是会被矮茶几绊倒在地并且弄得头破血流。很自然，电灯泡已经逐渐发展成和它们的用户之间的相互促进的联系。

问：但是为什么电灯泡不能有永久寿命呢？如果有的话，那不是可以使它们的用户的生活更好？

答：只有存在断成代谢，进化才会起作用。随着新的进步的出现，一些东西必须要消失。

问：在至关紧要的关头，复印机总是会中止十分钟。当印刷机器代替我的时候，我几乎失去了两份工作。它们肯定不是在保护它们的主要有机体，不是吗？

答：进化是一个错综复杂的平衡。一个有机体可以是很成功的，并且可以耗尽所有的生长环境。设想一个能干的复印机在一秒钟内进行了十亿次无错的复制，对复印机来说，每一个复制的东西都要是完美的、正确的，这可能吗？它是没有任何目的的，并且人们很快将用像一个装满啤酒的冷冻机那样的更有趣的东西来代替它。

问：说到啤酒，为什么有些易拉罐没有打开，其拉手就折断了呢？在一次钓鱼旅行结束的时候，我的冰箱里塞满了没有拉手的、不能打开的易拉罐。

答：你正在自问自答，不是吗？

问：为什么啤酒不能列入和人相互促进的联系当中？我当然可以和它共同生存。

答：在这种情况下，啤酒和人在竞争同样的生态环境。如果两个人喝啤酒，那么他们通常喝得酩顶大醉，好像变成另外一个人，而啤酒不会变。

问：当人们喝醉了的时候产生了另外一批啤酒和麦酒会是怎样呢？那些就是人与酒开始合作的标志吗？

答：进化是很难预测的。方程式里的一个小改变就会改变整个结果。我认为啤酒拉盖不久后将变得更难实现。为什么呢？这是因为有机体通常是不断再生地进化，以便能够抵制避免具有灾难性的竞争。你的方案将会很快地导致一场啤酒洪灾。

问：关于拉盖问题与我无关吗？进化论科学家就毫无价值吗？

答：进化是错综复杂的。如果科学家们能够回答所有的问题，那么就根本不需要进化论科学家的存在了，那些给进化论科学家发放资金的国家科学基金工作人员也不需要了，这里我们有一个定义明确的合作工作。

7.1 使用仿真语法

第6章显示了怎样隐藏数据和怎样把数据转变成一个文件的统计模式的仿真。例如，如果想要一张文本看起来像《纽约时报》，那么就要在纸张中插入大量的原材料并且聚集统计模式以使得仿真结果变得可能。理想上，这样一个函数应该有一个强大的技术，这种技术能够在自动扫描程序中隐藏数据，而这个程序是使用统计模式来确认数据的。

这些以霍夫曼为基础的方法，其输出无疑可以欺骗任何检查数据并寻找可疑模式的机器。字母应该符合所期望的分布，字母“e”是常见的，字母“z”是不常见的。如果第2或第3顺序文字被使用，那么“u”就应该跟在“q”后面，并且对一个只是检查统计值的计算机来说，每一件事好像都很有意义。

这些统计仿真函数不能够欺骗任何着眼于语法的人。像6.1节里发现的第1或第2顺序仿真看上去是无法理解的。虽然文字已经开始出现在第3或第4顺序，但是它们很少属于基本语法结构，甚至一个难以驾驭的语法检测器也只能把这些东西标记在一英里之外。本章描述了怎样产生语法上正确的仿真以及怎样使其有实际意义。这些算法的基础是一些语言学的基本著作，这种语言学现在支持很多计算机科学。最终结果只能被阅读，但是却非常非常难被破坏。

本章中使用的基本抽象概念是与上下文无关的语法。这个概念是由Noam Chomsky提出的，他以此来大略地解释语言是怎样工作的（参看[CM58]）。这个结构看上去更像句子图表的数学形式。这个模型被计算机科学家所采用，这些科学家不但探索它的理论限制，而且使用它作为像C或PASCAL那样的程序语言的基础。

一个与上下文无关的语法是由3个不同部分所组成：

- **终端** 这是一个有关用来整理得到最终结果的字或句子段落的技术术语。把它们看成印在迷惑段落上的模型。这些终端通常被称为字或短语。
- **变量** 它们是被用做稍后确定的抽象模型。它们和在代数学或程序设计里所用的变量非常相似。它们将用黑体字排版，像这样：**变量**。
- **句子成分** 描述了一个变量怎样被转化成不同的变量组和终端。

其格式如下：

变量 →单词‖短语

这意味着一个变量既可被转化成单词也可转化成一个短语。箭头（→）代表转化，双竖线（‖）代表“或”。在这个例子中，方程式的右边只代表终端，但也可以是混合变量。可以把产品作为迷惑部分如何相配的规则。

基本思想是一个语法描述一组被称做终端的字和一组关于它们怎样匹配的潜在的复杂规则。在很多情况下，在句子成分的每一个阶段，都有一个合适的位可供自由选择。

在这个例子里，变量既可被转化成字也可转化成一个短语。到底是转化成什么，这个选择发生在信息将被隐藏的地方。用很多方法可以使一个随机数字生成器驱动一个伪造的计算机化的诗歌机器，而同样的方法也可以使数据促使这个选择的决定。数据可以通过一个著名的语法分析的逆向过程被恢复。

这里有一个采样语法：

开始 →**名词 动词**
名词 →芬德‖ 布来尼‖芬德和布来尼
变量 →去钓鱼‖去玩保龄球

通过由变量“开始”并且应用句子成分传送不同的变量，这个语法就可以产生像“芬德和布来尼去钓鱼”这样的句子。这通常用一个箭头（→）写出，这个箭头表示几个不同句子成分的结合。像这样：开始→芬德和布来尼去钓鱼。陈述同一件事情的另外一个方法是这样的：句子“芬德和布来尼去钓鱼”。在由这个语法产生的语言里，句子成分的顺序是任意的，并且在某些情况下顺序可以是不同的（这不在这个简单的例子当中）。

更复杂的语法看上去像这样：

开始 →**名词 变量**
名词 →芬德‖ 布来尼
动词 →去钓鱼 地点 ‖去玩保龄球 地点
地点 →方向衣阿华州‖方向明尼苏达州
方向 →北部‖南部

为了简单，这个语法的每一个部分都有两种选择，我们把它们称为0和1。如果以变量“开始”作为开端总是处理最左边的变量，那么就可以从由这个语法产生的语言中把位转变成句子。一个逐步的过程阐述如表7.1所示。

表7.1 把1010转变成一个句子的5个步骤

步骤	进展中的回答	位隐藏	产生选择
1	出发	无	出发→名词 变量
2	名词 变量	1	名词→布来尼
3	布来尼 变量	0	变量→去钓鱼 地点
4	布来尼去钓鱼 地点	1	地点→在方向中
5	布来尼去钓鱼 方向 方向 明尼苏达州	0	方向→北部

位1010通过转变成句子“布来尼 去 钓鱼 在北明尼苏达”而得到隐藏。位0001将产生句子“芬德 去 钓鱼 在 南衣阿华州”，位1111将产生句子“布来尼 去 玩保龄球 在 南明尼苏达”。在由这个语法产生的语言中，有2^4个不同的句子，并且所有的句子意思都很清楚。

显然，复杂的语法可以产生复杂的结果，而要产生高质量的文字，则需要一定量的创造力。你必须预测字和短语怎样相配，还要确信每一个集合的部分都要措辞恰当。图7.1显示了一个扩展语法的结果，这个语法是由一场棒球比赛的画外音仿真发展而来的。完整的语法可以在附录B中找到。

```
Well Bob, Welcome to yet another game between the
Whappers and the Blogs here in scenic downtown Blovonia.
I think it is fair to say that there is plenty of
BlogFever brewing in the stands as the hometown comes out
to root for its favorites. The umpire throws out the
ball. Top of the inning. No outs yet for the Whappers.
Here we go. Jerry Johnstone adjusts the cup and enters
the batter's box. Here's the pitch. Nothing on that
one. Here comes the pitch. It's a curvaceous beauty. He
just watched it go by. And the next pitch is a smoking
gun. He lifts it over the head of Harrison "Harry"
Hanihan for a double! Yup. What a game so far today.
Now, Mark Cloud adjusts the cup and enters the batter's
box. Yeah. He's winding up. What looks like a
spitball. He swings for the stands, but no contact.
It's a rattler. He just watched it go by. He's winding
up. What a blazing comet. Swings and misses! Strike
out. He's swinging at the umpire. The umpire
reconsiders until the security guards arrive. Yup, got
to love this stadium.
```

图7.1　由附录B里与上下文无关的语法产生的一些文字

图7.1只显示了一个26K文件的第一部分，这个文件是由隐藏这样一个引用而产生的：

I then told her the key-word which belonged to no language and saw her surprise. She told me that it was impossble for she believed herself the omly possessor of that word which she kept in her memory and which she never wrote down...This disclosure fettered Madame d' Urfé to me . That day I became the master of her soul and I abused my power.——*Casanova, 1757, as quoted by David Kahn in the Codebreakers [Kah67]*

语法在很大程度上依赖于提供最终结果形式的棒球比赛的结构。因为语法的建立需要小心谨慎，所以球、击打和出局的数字必须保证精确。而另一方面，因为语法不能追踪到它们的路径，所以跑动的数字就被省略了。这可以很好地阐述可更改的“与上下文无关”是什么意思。应用于一个特殊变量的部分并不依赖于围绕变量的上下文。例如，在刚才的基本例

子中，芬德和布来尼谁去钓鱼和谁去打保龄球，这并不是问题，做出地点是在明尼苏达州还是在衣阿华州的决定是独立的。

附录B里的“棒球语法”在每一半局里都使用了一个单独变量。一个半局可能以产生一串陈述每一个人的本垒打跑的句子而结束。这个信息及其上下文并不受下一个半局里句子成分选择的影响，这只是因为变量和句子成分被定义的方法所增加的一个限制。如果产品不是那么随意，并且基于更多的计算上，那么更好的文字就可以产生①。

7.1.1 语法分析和回顾

由一个特殊的语法产生的句子隐藏信息是一个漂亮的玩具，句子中的数据把室内游戏转变成隐秘地传输信息的一个真正工具。逆向过程被称为语法分析，计算机科学家广泛地研究了这个过程。像C语言那样的计算机语言是由一个与上下文无关的语法建立的，计算机为了了解语言指令而对语言进行分析。这一章主要讲的是把句子反向转变成引向其产品的位清单的过程。

语法分析可能是复杂的，也可能是容易的。大多数计算机语言都被设计得使语法分析很容易，这样处理过程就会很快。没有原因能够解释为什么仿真不能这样做。可以一直分析任何与上下文无关语法的句子并且恢复产品的顺序，但是为什么要使用这些任意的复杂程序，这也是没有原因可解释的。如果这个语法被正确地设计，那么对任何人来说，分析数据都是足够容易的。

下面有两种关键规则。第一，确信语法不是含糊的；第二，保持语法在Greibach Normal Form（格里巴克范式，简称GNF）里。如果相同的句子通过两组不同的产品从一个语法中出现，那么该语法就是“含糊”的。因为没有什么方法能够准确地恢复数据，所以这使得语法对隐藏信息是没有用的。一个含糊的语法作为一个可爱的诗歌生成器是有用的，但如果没有什么方法能确保隐藏的含义是什么，那么它就不能被用来隐藏信息。

这里有一个含糊语法的例子：

> **开始** →名词　动词 ‖ 谁　做什么
> **名词** →芬德 ‖ 布来尼
> **动词** →去钓鱼 ‖ 去玩保龄球
> **谁** →芬德去 ‖ 布来尼去
> **做什么** →玩保龄球 ‖ 钓鱼

句子“芬德 去 钓鱼”可以由两个不同的步骤产生。如果正在句子中隐藏信息，那么“布来尼 去 玩保龄球”既可以从位011也可以从位110得到。这样一个问题无论如何都要避免。

如果一个与上下文无关语法在GNF之中，那么它就意味着变量是在产品的末尾。这里有一些例子：

①产生一个可在特殊时间里更好地编码的复杂语法是可能的，但这个语法并不是完美的。它可以把工作完成得更好，但它不一定很准确。这作为练习留给大家。

产品	是否在GNF
开始 →名词　动词	是
地点 →在 方向 衣阿华州 ‖ 在 方向 明尼苏达州	否
地点 →在 方向　开始 ‖ 在 方向 开始	是
做什么 →玩保龄球 ‖ 钓鱼	是

把任何任意的与上下文无关的语法转变成GNF是很容易的。可以只是简单地增加产品直到成功为止。这里是本节中用一个新的变量“开始”的扩展例子，它是这样放在GNF里的：

开始 →名词　动词
名词 →芬德 ‖ 布来尼
动词 →去钓鱼 地点 ‖ 去玩保龄球 地点
地点 →在 方向　开始
方向 →北部 ‖ 南部
开始 →衣阿华州 ‖ 明尼苏达州

这个程序是一个模仿工具，无论遇到什么样的上下文，它都被设计用来吸收地方特色，并且作为一个应急的优先级顺序的超越代表其本身。

——Willian Gibson，Burning Chrome。

这个语法准确地产生了同样的句子组或语言作为另外的模型，惟一的不同之处是它们被选择的顺序。这里，当变量“地点”被固定时，就没有什么可利用的选择了。在这一个点没有位可以被储存起来。变量“方向”和“开始”将按顺序被处理，其结果就是句子“布来尼去明尼苏达州北部钓鱼”，它是由位1001产生的。在前面表7.1阐述的语法中，该句是由隐藏位1010而出现的。

从一个在GNF里的与上下文无关语法分析出结果一般来说是很容易的。表7.1显示了句子“布来尼去明尼苏达北部钓鱼”是怎样由位1010产生的。这个语法分析过程是沿着相似的线路完成的。表7.2显示了使用上面GNF里的语法分析句子的过程。

表7.2　分析一个句子

步骤	问题中的句子片断	匹配的产生	恢复的位
1	布来尼去明尼苏达州北部钓鱼	名词→芬德 ‖ 布来尼	1
2	布来尼去明尼苏达州北部钓鱼	变量→去钓鱼 地点 ‖ 去玩保龄球 地点	0
3	布来尼去明尼苏达州北部钓鱼	地点→在方向 出发	无
4	布来尼去明尼苏达州北部钓鱼	方→北部 ‖ 南部	0
5	布来尼去明尼苏达州北部钓鱼	出发→衣阿华州 ‖ 明尼苏达州	1

位1001在第5步里被恢复。这显示了一个简单的语法分析过程是怎样恢复位的，而这些位储存在使用GNF里语法产生的句子里面。更好的语法分析算法能够处理任何任意的与上下文无关的语法，但是这已在这本书所讨论的范围之外。

7.1.2 优越性

测量优劣有很多种方法，但是这里最重要的一种方法是有效的和能够抵制攻击的方法。这种方法的有效性在某种意义上很大地依赖于语法本身。在本节的例子里，源文本的一位被转变成像“明尼苏达州”或“布来尼”这样的单词，那不是特别有效。

如果有更多的选择的话，那么这个语法就可以在句子形成的每一个阶段编码更多的位。在每一个例子里，句子成分的右边都只有两个选择。没有原因能解释为什么不能有更多的选择。4个选择应该可以编码两个位，8个选择可以编码3个位，如此等等。更多的选择通常是很难添加的。为什么不能由句子的名词产生一个有1024个名字的人，这也是没有原因可解释的。在一次摄取中将编码10个位，惟一的局限就是想像力。

估计对攻击的抵制是更复杂的，最难的测试可以欺骗一个人。第6章里产生的文字从统计学来说看上去是正确的，但是甚至是最好的第5顺序文字，对一个普通人来说看起来也是愚蠢的。从这个过程中产生的语法文字和创造这个语法的人一样，是令人信服的。附录B中的例子显示了它会有多么复杂。努力花几天时间在一个语法上是很值得的。

这种形式仍然有局限，与上下文无关的语法具有一种非常简单的形式，尽管如此，这还是意味着它们不能真正专门地追踪信息。附录B中的例子显示了击打、球、出局是怎样保持直线性的，但是它却不能保持对比分和棒球运动员移动的追踪。实质上，一个更复杂的语法就是从此开始的，但是在这个格式里写字的话，总是会有限制的。

与上下文无关的特性也把更深层次的问题强加在了注解语句上。因为故事发现自己不断重复地处在同样的情形下，所以一场棒球比赛的画外音在这里是一个突发奇想。击球手面对着投手，比分的细节和记分在变化，但是比赛过程却是在不断地重复再重复。

建立一个能产生令人信服的结果的语法可以说很容易，也可以说是很难的。其难度在很大程度上是依赖于冷嘲热讽的水平。例如，任何人都可以很容易地证明美国华盛顿政府所执行的功能过程需要3个步骤：

1. 议员X预示要改变企业Z的规章制度Y。
2. 企业Z为了议员P，D，Q的重新竞选而被迫付出金钱。
3. 议员P，D，Q在委员会里停止X的计划。

如果相信在华盛顿的生活大概就是这样的一个经济过程，那么提出一个很长的、复杂的且能够扩张来自于华盛顿的新闻的语法，对你来说应该是没有问题的。对肥皂剧或其他被蒸馏的生活本质也可以同样这样说。

对于语法要由什么样的数学攻击类型构成，这里也有一些更深层次的问题。任何想恢复位的攻击者都必须了解用来产生句子的语法的一些情况。对传输双方来说，这都需要被保密。算出能产生一个特殊句子组的语法是不容易的。7.1.1中的第一个例子，即那个更含糊的语法例子，显示了5个简单的构成规则是怎样用两种方法产生大量的句子的。每一个句子都可能由很多不同的语法产生，在实践中把所有的语法找出来是不可能的。

Alicebot方案使得计算机以自然的语言交谈。设想如果它们都在同一时刻编码信息，会怎样呢？参看www.alicebot.org。

重新建立语法也不是特别可行的。决定由一个变量结束而产生的字的位置和另一个变量开始而产生的字的位置是一个很困难的任务。当发现有一个句子类型在不断地重复的时候，也许就可能产生这样一个推断。这些理由无论如何并不能保证系统的安全性。7.2.4节“仿真理论的安全性评估”，讨论了一些信任系统安全性的更深层的原因。

7.2.3节“加密语法”，显示了怎样重新安排语法以得到更好的安全性。

7.2　创建以语法为基础的仿真

产生一个执行与上下文无关仿真的软件并不是很复杂的。只需要对怎样分析文字、产生一些随机数字、把数据分解成位层有一个基本的了解。附录A显示了一个完整的、可编码和解码信息的、与上下文无关语法系统的Java代码。

一个C语言代码版本在代码光盘上也是可利用的。这是一个非常直接的转变。

有很多不同的代码细节需要解释。最好的开始是语法文件的格式。图7.2显示了在整个附录B里描述的与上下文无关语法的一小部分。

```
*WhapperOutfieldOut = He pops one up into deep left field./.1/
      He lifts it back toward the wall where it is caught
        by *BlogsOutfielder *period/.1/
      He knocks it into the glove of
        *BlogsOutfielder *period /.1/
      He gets a real piece of it and
        drives it toward the wall
        where it is almost ... Oh My God! ... saved by
        *BlogsOutfielder *period /.1/
      He pops it up to *BlogsOutfielder *period /.2//

*WeatherComment = Hmm . Do you think it will rain ? /.1/
  What are the chances of rain today ? /.1/
  Nice weather as long as it doesn't rain . /.1/
  Well, if rain breaks out it will
    certainly change things . /.1/
  You can really tell the mettle of a
     manager when rain is threatened . /.1//

*BlogsOutfielder = Orville Baskethands /.1/
                   Robert Liddlekopf /.1/
         Harrison "Harry" Hanihan /.1//
```

图7.2　附录B所述语法的3个部分在被代码承认的文件格式里被编码

变量是以星号开始而且必须是一个邻接字。一个更好的编辑器和分析器的结合能够把它们区分开来并且移除这个限制。以一个很像星号的伪造字符作为开始，是最好的折衷办法。虽然它减小了可读性，但是它却保证了没有任何模糊性。

在每一个变量出现的产品列表被前面的一个斜线隔开，模式是**phrase / number/**。由一个变量产生的最终短语在最终的数字后有一条额外的斜线。这个数字是一个给定随机数据选择发生器的加权平均值。在这个例子里，这个数字大多数都是 .1。这个软件把所有的加权平均值加起来并形成一个特殊的变量，并且通过这个总数划分使得选择规格化。

加权平均值并不是随意使用的。如果一个特殊短语的选择将要编码信息，那么在输入位和输出位之间必须有一个一对一的连接。在6.1.1里讨论的霍夫曼树"选择下一个字母"，是把一个加权的选择映射到输入位的最好方法，所以加权值被用来建立一棵树。图7.3有一棵这样的树，它在制作一个项目（译者：指棒球）的外场手Blogs的选择里隐藏信息。同样的证据证明了霍夫曼树是压缩文件和编码信息的最好方法。

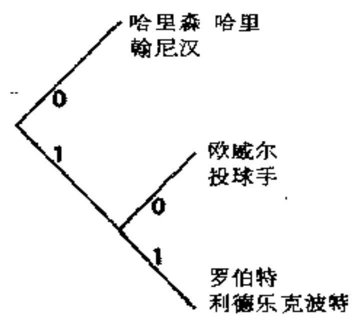


图7.3　霍夫曼树被用来在制作一个特殊项目（译者：指棒球）的外场手Blogs的选择里隐藏信息

自然，霍夫曼树只是近似所期望的统计结果，并且近似值的水平受到一半的幂的限制。图7.3显示了霍夫曼树的不相关性有多么严重。其中一个选择编码一个信息位，另外两个选择则编码另外两个信息位。这实际上意味着，第一个选择将占有50%的时机，而另外两个选择将各占25%。

随着越来越多的选择可利用，不精确的程度在下降。例如，这应该是很显然的：如果一个变量可以被转变成2^i个不同的选择，且每一个选择都等于加权值，那么近似值将是很接近的。也可能是这种情况：如果所有的加权值都是2的幂并且总数加起来也是一个2的幂——例如{1，1，2，4，2，2，4}。

7.2.1　输出的语法分析

仿真语法分析器里的代码，其作用是处理把仿真值回传给产生仿真值的位。分析一个与上下文无关的语法输出是一个很好理解的问题，并且在计算机科学文献里被深入地讨论。最好的语法分析器，可以把一个语法的任何文字按顺序转变回导致产生文字的句子成分。像CYK算法（参看[HU79]）那样的最普通的算法是比较慢的。

在这个代码里执行的语法分析算法是一个折衷方法，它的运算对象只是GNF里一个限制版本的语法。这个形式需要把任何变量都放在句子的末尾。7.1.1中展示了一些例子。因为需要使通过检查一个句子的第一个字来决定做出选择变得简单，所以这个语法分析器要求的形式是更严格的。这意味着同一个变量的两个选择前n个字都不能相同，变量n是可调整的，但是它越大，算法就会变得越慢。

因为语法分析器只需考虑单词和短语，所以这个格式实质上使得语法分析变得更加容易了。跟踪变量和做出猜测是没有必要的。阐述这个的最好方法是用一个不遵循此规则的语法。这里有一个在不正确格式下的语法：

> **开始** →名词　动词
> **名词** →芬德　与朋友一起 || 芬德　自己
> **与朋友一起** →和布来尼去钓鱼　地点 || 和布来尼去玩保龄球　地点
> **自己** →去钓鱼　地点　|| 去玩保龄球　地点
> **地点** →在　方向　开始
> **方向** →北部 || 南部
> **开始** →衣阿华州 || 明尼苏达州

设想面对这样一个句子："芬德和布来尼去衣阿华州北部钓鱼"，这是由位/选择0000（译者：原文是bits/choices 0000）产生的。分析这个句子并且恢复位当然是有可能的，但是这并不容易。成分"名词→芬德　与朋友一起 || 芬德　自己"并不能使人很容易决定做出哪个选择。每一个选择开头的结尾词是相同的，它们两个都是"芬德"。一个语法分析器需要检查扩展变量与朋友一起和自己的结果来决定应该选择哪一条路径，跟随这些途径是可行的，但是它会使算法变慢并且使结果变得复杂。大多数严格的语法分析器都能够处理这个问题。

这个设备在这方面是迟钝的，但是作者不认为是作牺牲的一击（译者：指棒球）。把语法放在正确的格式里相对是比较容易的，这个语法应该被修改如下：

> **开始** →名词　动词
> **名词** →芬德和布来尼　做什么 ||　芬德　做什么
> **做什么** →去钓鱼　地点　|| 去玩保龄球　地点
> **地点** →在　方向　开始
> **方向** →北部 || 南部
> **开始** →衣阿华州 || 明尼苏达州

任何与上下文无关的语法都可以放在GNF里，安排GNF里的语法扩展以便消除模糊性也是有可能的。唉！有时n需要被设成很大的值才能完成这个任务。另外一种解决方法是采取更复杂的语法分析算法。

7.2.2 建立语法的建议

如果想建立一个能被用来有效地把数据转变成无碍的文字的语法，那将是一个很耗时的过程。更多的字和短语意味着更多的选择，而更多的选择意味着更多的数据可被打包进入位置。具有很长的短语和很少的选择的语法可能是没有效的。这里有一些建议：

David McKellar建立了一种语法，此语法能在从他的邮件信息集合删除的类似于邮件的短语里隐藏信息。可以参看网页www.spamming.com。

关于图表和注解语句的思考 在附录B里的一场棒球比赛画外音使用的语法，是一个能成功地保持有充足重复风格的典型例子。因为重复效应可以节省很多的努力，所以这样的风格是简单的与上下文无关语法的最好选择。为了使系统工作，没有必要一个接一个地产生句子。同样的选择可以重复地被使用。

还有其他一些好的领域可供探索。股票市场分析一般来说是与上下文无关的，并且充满了一堆在全世界流动的意识流的杂乱数字。没有人能总结出为什么会有成千上万的人不断地买股票和卖股票。还有，体育报道经常提出不同的方法来进行总结，如“X撞击了Y”或“X使Y停止”。为什么不能建立一个更完善的版本，使其可供给实际的新闻以修改语法，以便保证数据的正确和填满隐藏的位？这也是没有原因可解释的。

还有其他一些自然与图表无关的领域。比如，现代诗和不受格律约束的自由诗体采用了超乎寻常的风格，人们看了上一句并不知道下一句会是什么。一个由拙劣设计而得的语法，所产生的一个奇怪的东西并不能持续多久。另外，人的大脑能在任意的位置熟练地找到模式及其意思，实际上人类就可以涉人由此产生的工作。

分解句子 有更多的选择存在，就有更多的数据可被编码。为什么每一个句子不能分解成名词短语、动词短语、宾语短语，这是没有原因可解释的。很多句子都是以感叹词和讲道词开始的，我们应该使它们变得多样化。

使用多种变化 更多的选择意味着更多的数据被隐藏。述说同一件事情可以有很多种不同的方法，同样的想法可以用一千种不同的形式来表达。一个优秀的作家能够重复讲述一个相同的故事，为什么要止于一个简单的句子呢？

7.2.3 加密语法

建立一个复杂的语法是不容易的，所以，如果这个语法能够重复地被使用，它就是理想的。自然地，当同样的模式在加密中被重复时，会有很多问题出现。这给了攻击者另外一个寻找相似处，以及模仿并破坏系统的机会。在建立语法的过程中，大多数工作是捕获人们交流的正确风格，把单词和短语安排形成句子实际上并不是那么重要。例如，上面所述的几种产生关于芬德和布来尼的句子的语法，它们实际上是产生同样的句子集合，即使这些语法是不同的。有很多不同的语法可以产生同样的语言，为什么语法不能自动地转变成不同的版本模式，这是没有原因可解释的。

这里描述了三种主要的转化：扩写、缩写和置换。

扩写 一个句子中的一个变量在另一个句子中用所有可能的方法被扩写，这就像代数学当中的分配率，例如：

名词 →芬德 与朋友一起 ‖ 芬德 自己
与朋友一起 →和布来尼去钓鱼 地点 ‖ 和布来尼去玩保龄球 地点
自己 →去钓鱼 地点 ‖ 去玩保龄球 地点

. . .
. . .
. . .

第一个变量，与朋友一起，通过建立一个所有可能的含有名词的句子被扩写，然后含有“与朋友一起”的句子就从这个语法中消失：

名词 →芬德和布来尼去钓鱼 地点 || 芬德和布来尼去玩保龄球 地点
|| 芬德 自己

自己 →去钓鱼 地点 || 去玩保龄球 地点

. . .

. . .

. . .

缩写 缩写和扩写相反。如果在几个句子里有某个相同的模式，那么它就可以被一个新的变量所代替，模式“芬德和布来尼”可以在两个含有名词的句子里被找到：

名词 →芬德和布来尼去钓鱼 地点 || 芬德和布来尼去玩保龄球 地点
|| 芬德 自己

自己 →去钓鱼 地点 || 去玩保龄球 地点

. . .

. . .

. . .

这可以通过引入一个新的变量做什么而被缩写：

名词 →芬德和布来尼 做什么 地点 || 芬德 自己

做什么 →去玩保龄球 || 去钓鱼

自己 →去钓鱼 地点 || 去玩保龄球 地点

. . .

. . .

. . .

这个新的语法与扩写过程开始的语法不同。它产生了同样的句子，但是却有不同的位模式。

置换 句子的顺序可以被打乱。这在任何已经建立的霍夫曼树里面可以改变它们的位置，加密也可以取代树本身。

任何扩写、缩写和置换的组合将产生一个能生成同样语言的新的语法，但是这个新的语法将以完全不同的方式从位里产生句子。这增加了安全性，并且使得任何攻击者能推断出有关语法的连贯信息的可能性下降了。

这些扩写、缩写和置换是由一个密钥控制的伪随机数字生成器产生的，交谈双方的每一个人都可以用同样大的语法开始，然后通过输入话路密钥，他们可以使双方的随机数字生成器同步。如果这个随机数字生成器指导扩写、缩写和置换语法的过程，那么交谈两端的语法都将保持相同。在一个预先确定的变化数量后，结果就会被冻结在某一个位置。虽然双方都仍然有同样的语法，但是实质上从语法开始的时候起它们就不同了。如果每一次都这样做，那么结构就会有较大的不同，这样攻击者就更难破坏系统了。

这里有一个扩写、缩写和置换的更仔细的定义。已知与上下文无关的语法G，而成分由$A_i \rightarrow a_1 \| a_2 \| ... \| a_n$组成，$A_i$是变量，$a_j$是成分，它是终端值和变量的一个混合。

扩写采取如下步骤：

1. 选择一个包含A_i的成分，其形式是：$V \rightarrow \beta_1 A_i \beta_2$。$V$是一个变量。$\beta_1$和$\beta_2$是终端值和变量的串。
2. 我们说这个A_i可以被n个成分代替：$A_i \rightarrow a_1 \| a_2 \| ... \| a_n$。选择这些成分的一个子集并且称它为$\Delta$。而一组成分不是称$\Delta$，而是称做$\bar{\Delta}$。
3. 对选择的每一个A_i的成分，用另外一个V的成分代替形成$V \rightarrow \beta_1 a_i \beta_2$。
4. 如果所有的成分都被扩写（即，$\bar{\Delta}$是空的），那么就从V的成分组中删除$V \rightarrow \beta_1 A_i \beta_2$。否则，就用成分$V \rightarrow \beta_1 A_k \beta_2$代替它，这里$A_k$是由于成分被拉出$V \rightarrow \beta_1 A_k \beta_2$而引入系统的新变量。那就是说，对所有在$\bar{\Delta}$的$A_k$，有$A_k \rightarrow a_i$。

要注意所有的成分都必须被扩写。语法尺寸的影响很难被预测，如果a_i含有n个成分，并且变量本身是在m个多样的、不同的其他变量的右边被找到的，那么一个彻底的扩写将产生nm个成分。

一个缩写由以下几个步骤完成：

1. 找到一组字符串$\{\gamma_1 ... \gamma_n\}$，使其对每一个$\gamma_i$都存在满足$V \rightarrow \beta_1 \gamma_i \beta_2$的成分。$\beta_1$和$\beta_2$是终端值和变量的集合。
2. 产生一个新的变量A_k。
3. 对每一个i产生$A_k \rightarrow \gamma_i$。
4. 对每个i删除成分$V \rightarrow \beta_1 \gamma_i \beta_2$，并且用$V \rightarrow \beta_1 A_k \beta_2$代替它们。

要注意，所有的成分都必须缩写。如果它被成功地应用的话，语法就可以较大地被缩短。

扩写和缩写操作是很有效的。如果两个语法G_1和G_2产生同样的语言，那么就会有一些扩写和缩写的混合将把G_1转变成G_2。因为扩写操作会一直到没有什么东西可扩写时才会停止重复，所以这是很容易明白的。整个语法是由一个开始符号和一个从语言中把开始符号转变成一个句子的成分组成。对语言里的每一个句子，都有一个变量和一个句子成分。有一个扩写表可以把G_1和G_2都转变成同样的语言，这个扩写表可以通过一组倒置的逆向缩写而得到。把G_1转变成G_2也是一样，简单地扩写整个G_1然后应用与扩写相反的缩写就可以扩写G_2了。因为一个语法的完整扩写是非常大的，所以，这个证明可能永远不会在实践中被使用。

扩写和缩写的最重要的作用，是它怎样重新安排正在被编码的位之间的联系和句子的结构。这里是一个语法范例：

名词 →鲍勃和雷蒙德 动词 ‖ 芬德和布来尼 动词 ‖
莱温妮和雪莉 动词 ‖ 德尔玛和露丝 动词
动词 →去钓鱼 地点 ‖ 去射击 地点 ‖ 去飞行 地点 ‖ 去蹦极跳 地点
地点 →在明尼苏达州 ‖ 在马里 ‖ 在加德满都 ‖ 在喀拉哈里沙漠

这些变量每一个都有4种选择。如果它们的加权值相等，那么我们就能用每一个选择编码2位。按顺序把数字00，01，10和11列出。所以隐藏位110100产生句子“德拉和路易丝 去射击 在明尼苏达。”

图7.4显示了把12个短语转变成位的方法。

这里也有一个模型。隐藏的位010100产生句子“芬德和布来尼 去射击 在明尼苏达”前两个位直接与句子的名词相关，第二个两位与句子的动词相关，第三个两位与位置相关。大多数人建立一个语法都会跟着一个相似的模型，这是因为它遵从结构的自然印象。因为一个攻击者可能足够聪明到可利用这个模型，所以这种语法（跟着一个相似的模型）是很危险的。按顺序的扩写可以修补它。这里是发生了几个变化后的语法：

名词 → 鲍勃和雷蒙德 动词2 || 芬德和布来尼 动词4 ||
莱温妮和雪莉 动词 ||
德尔玛和露丝 动词3 ||
鲍勃和雷蒙德去钓鱼 地点 ||
鲍勃和雷蒙德去射击 地点 ||
德尔玛和露丝去钓鱼 地点 ||
德尔玛和露丝去蹦极跳 地点 ||
芬德和布来尼去明尼苏达州射击 ||
芬德和布来尼去马利射击 ||
芬德和布来尼去加德满都射击 ||
芬德和布来尼去喀拉哈里沙漠射击

动词 → 去钓鱼 地点 ||
去射击 地点 ||
去飞行 地点 ||
去明尼苏达州蹦极跳 ||
去马里蹦极跳 ||
去加德满都蹦极跳 ||
去喀拉哈里沙漠蹦极跳

动词2 → || 去飞行 地点 || 去蹦极跳 地点

动词3 → 去射击 地点 ||
去飞行 地点

动词4 → 去钓鱼 地点 || 去飞行 地点 | 去蹦极跳 地点

地点 → 在明尼苏达州 ||
在马里 ||
在加德满都 ||
在喀拉哈里沙漠

变量名词的成分已经用很多种方法被扩写了。一些也已经扩写了变量动词并且使它们完全地合为一体，而其他的却只有结合的一部分。现在，变量动词有4个不同的版本，它们是因为处理没有被扩写的部分而建立的。

缩写的作用是越发明显的。图7.4显示了霍夫曼树是怎样把位转变成变量名词的成分的。名词、动词、位置和产生它们的位之间的联系现在更加难以被察觉了。前两个位现在已经不再与名词相对应了。

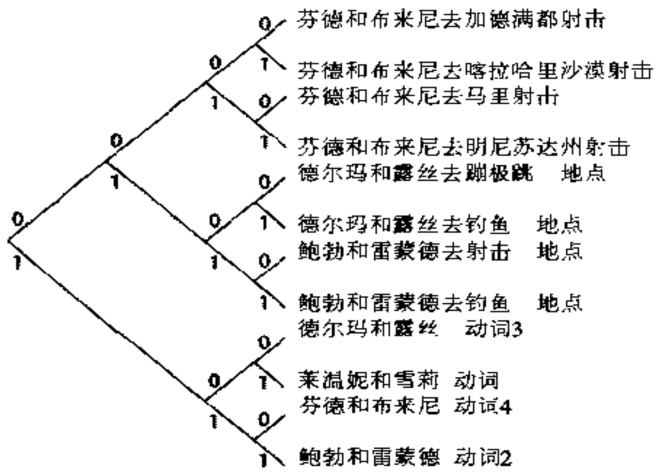


图7.4 一个把位转变成变量名词的成分的霍夫曼树

表7.3显示了一些短语和产生它们的位[①]。

句子之间仍然有一些相关性。表7.3里的前两个句子结尾是不相同的，这反映在最后两位上。事实上，前两位看起来好像意思是“芬德和布来尼 去射击”，而最后两位是选择位置。如果句子成分的顺序被倒置，那么这个模型就很容易被消除。它也会受到任何给定的短语的加权值的影响。

表7.3 6个句子以及建立它们的位，后面的图7.6显示了这个语法的一棵霍夫曼树

短语	位
芬德和布来尼去加德满都射击	0000
芬德和布来尼去明尼苏达州射击	0011
芬德和布来尼去明尼苏达州钓鱼	110100
芬德和布来尼去明尼苏达州蹦极跳	1100100
德尔玛和露丝去明尼苏达州蹦极跳	010000
德尔玛和露丝去马里飞行	100101

但是同样的模型对其他句子并不适用。如果句子以“芬德和布来尼去钓鱼”开始，或者如果他们是去“蹦极跳”，那么一个不同的模型就成立了。位置是由当变量地点被扩写时做出的选择所决定的。在这个情况下，位和位置之间的联系是不同的。这种情况下“明尼苏达”是由位00产生的。

这显然是阐述了这个结果是所有系统安全性的基础。短语“明尼苏达”的意思依赖于上下文，在大多数情况下它是由位00产生的，但是在少数情况下它也可由位11产生。因为语法被称做“上下文”，所以从某种意义上说这是具有讽刺意味的。这个术语是正确的，但是语法的结构仍然会影响结果。

①位和句子成分之间的一些联系是不能解释的，并且读者是很难发现的。

对系统安全性的更深入探索可以在7.2.4节里找到，即“仿真理论的安全性评估”。

缩写过程会带来更多的混淆。这里是一个经过几次缩写后的前面的语法：

名词 → 鲍勃和雷蒙德 动词2 ||
芬德和布来尼 动词4 ||
莱温妮和雪莉 动词 ||
德尔玛和露丝 动词3 ||
谁 去钓鱼 地点 ||
鲍勃和雷蒙德去射击 地点 ||
德尔玛和露丝去蹦极跳 地点 ||
芬德和布来尼去明尼苏达州射击 ||
芬德和布来尼去马里射击 ||
芬德和布来尼 动词5

谁 → 鲍勃和雷蒙德 || 德尔玛和露丝

动词 → 去钓鱼 地点 ||
去射击 地点 ||
去飞行 地点 ||
去明尼苏达州蹦极跳 ||
去喀拉哈里沙漠蹦极跳 ||
去蹦极跳 地点2

动词2 → || 去飞行 地点 ||
去蹦极跳 地点

动词3 → 去射击 地点 || 去飞行 地点

动词4 → 去钓鱼 地点 ||
去飞行 地点 ||
去蹦极跳 地点

动词5 → 去加德满都射击 ||
去喀拉哈里沙漠射击

地点 → 在明尼苏达州 || 在马里 ||
在加德满都 || 在喀拉哈里沙漠

地点2 → 在马里 || 在加德满都

两个新的变量动词5和地点2通过一个简单的缩写被引入。它们将很大地改变位和几个句子选择之间的联系。图7.5显示了新的几种霍夫曼树，它们被用来把位转变成句子变量成分的选择。表7.4显示了一些句子和产生这些句子的位在缩写前和缩写后的状态。

名词、动词和地点之间的一些联系仍然被保持，但是在某些方面有一些大的改变。一系列的扩写和缩写可以足够把任何语法打乱，并毁坏这些联系中的任何一个。

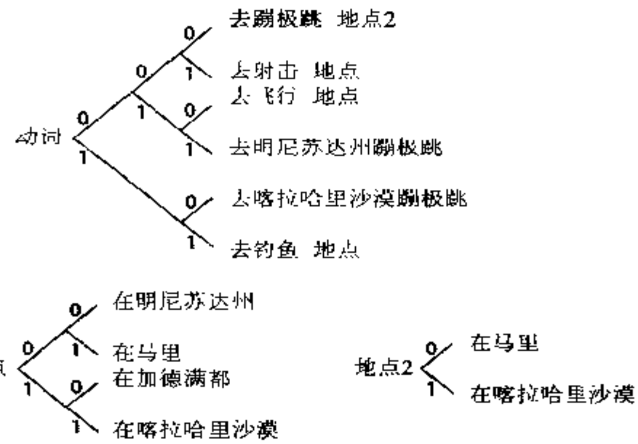


图7.5 一棵把位转变成表7.3中变量成分动词、地点和地点2的霍夫曼树

表7.4 5个句子以及产生它们的两组不同的位

短语	缩写前的位	缩写后的位
莱温妮和雪莉去明尼苏达州蹦极跳	101100	101011
莱温妮和雪莉去马里蹦极跳	101101	1010000
芬德和布来尼去喀拉哈里沙漠射击	0001	01111
芬德和布来尼去明尼苏达州蹦极跳	1100100	1100100
德尔玛和露丝去明尼苏达州蹦极跳	010000	010000

第三个变量“谁”，通过缩写也被引入，但是它产生了一个不是在GNF里的句子成分（名词→谁 去钓鱼 地点）。

在附录A里所示的执行过程里，这是不能由语法分析器做到的，但是用一个更好的语法分析器的话，将工作得很好。语法仍然是模糊的。这个例子只包括不在GNF里的语法附近工作的扩写和缩写显示。

一个有趣的问题是，在这种情况下，被应用到句子成分中的位是什么样的顺序。前面的在GNF里的例子使用了这个规则，即最左边的变量总是按顺序被扩写的。这个规则在这里很适用，但是它却导致了一个令人感兴趣的问题：重新安排。在GNF例子里，句子的第一部分总是和第一位相对应。在这个情况下，规则就失效了。表7.5显示了一个假定最左边规则的步骤。

表7.5 最左边的句子成分

开始	位的选择	句子成分
名词	0010	谁 去钓鱼 地点
谁 去钓鱼 地点	0	鲍勃和雷蒙德去钓鱼 地点
鲍勃和雷蒙德去钓鱼 地点	11	鲍勃和雷蒙德去喀拉哈里沙漠钓鱼

为什么句子成分的顺序必须和一个最左边的第一规则相关，这是没有原因可解释的。普通的语法分析算法在一个句子的产生中能够发现3个不同的选择。受GNF限制的语法分析器则不能处理这个问题，句子成分可以以任何预先定义的顺序被安排，该顺序是被交流链接的双方所使用的。所以句子“鲍勃和雷蒙德去喀拉哈里沙漠钓鱼”可以说是由6个组合的任意一个产生的，这6个组合是：0010011，0010110，0001011，0110010，1100010，1100100。

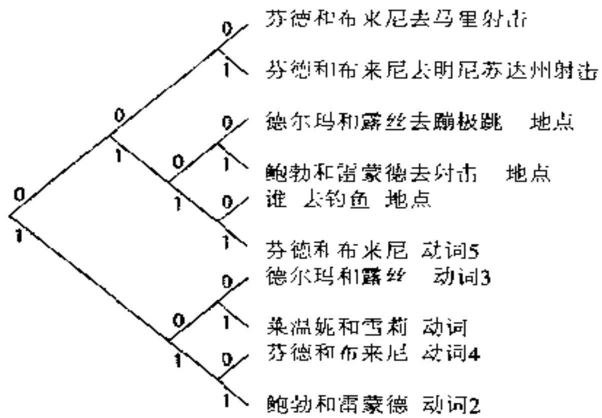


图7.6 一棵霍夫曼树，它可以把位转变成表7.3里的变量名词的句子成分

关于扩写和缩写的最后一节忽略了这样一个功能，那就是允许一个用户根据预先定义的议程求得选择的加权值，这些加权值可以通过扩写和缩写过程被准确地携带。如果有一个扩写，那么术语就会按比例放大。如果有一个缩写，那么术语就会聚集在一起。这里是一个扩写的例子，加权值作为变量在圆括号里显示。

扩写前：

名词 → 德尔玛和露丝 做什么（a_1） ||
哈里和露丝 做什么（a_2）

做什么 → 去射击（a_3） || 去医院 （a_4）

扩写后：

名词 → 德尔玛和露丝 做什么（a_1） ||
哈里和露丝去射击（a_2a_3 / ($a_3 + a_4$)） ||
哈里和露丝去医院（a_2a_4/ ($a_3 + a_4$)）

做什么 → 去射击（a_3） || 去医院（a_4）

下面还有一个同样的例子的缩写。

缩写前：

名词 → 德尔玛和露丝 做什么（a_1） ||
哈里和露丝去射击（a_2） ||
哈里和露丝医院（a_3）

做什么 → 去划船（a_4）‖
　　　　去钓鱼（a_5）

缩写后：

名词 → 德尔玛和露丝 做什么（a_1）‖
　　　　哈里和露丝 做什么2（$a_2 + a_{13}$）
做什么 → 去划船（a_4）‖
　　　　去 钓鱼（a_5）
做什么2 → 去射击（$a_2 / (a_3 + a_2)$）‖
　　　　去医院（$a_3 / (a_3 + a_2)$）

这些规则可以被任意地扩张以便处理所有的扩写和缩写。像圆括号里的加权值很大地影响了使用霍夫曼树把位转变成短语的方法。霍夫曼树只有在加权值的构成是正确的情况下才能很好地工作，所以很有可能大多数的树会产生有较大偏差的加权值的近似值。当扩写和缩写改变了树的结构时，加权值就将改变产生的模式。

7.2.4 仿真理论的安全性评估

确定以与上下文无关的语法为基础的仿真函数的强壮性并不是一件很容易的任务。有两个基本的方法可以完成，而这两个方法都会令人怀疑。第一个方法是在理论上分析系统的结构，并且将这个系统和其他的系统相比较。这暗示了要破坏这些系统里的仿真值是很困难的，但是并不能证明这就没有漏洞了。只是知道在过去，其他人试图破坏相似的系统并且失败了。这是一个很好的消息，但并不是结论性的。在这些以语法为基础的仿真函数（这些仿真函数在其他的系统中很难被使用）里，可能会有一些新的漏洞很容易被利用。

> 对你来说，主要的困难在于没有足够多的证据这个事实。重要的东西被一些不相关的东西覆盖和隐藏了。
>
> ——Arthur Conan Doyle，*Naval Treaty*

在理论方法中漏洞是很普通的。例如，有很少证据可以显示解决一个数学问题有多难。分类的数字是少量例子中的一种。如果有n个数字的一个列表，那么$cn\log n$是按时间比例的，这里c是某个基于机器的恒定值（参看[AHU83]）。这是一个很不错的结果，但是它却不能作为一个加密安全系统的很好的理论基础。还有其他一些分类算法能成功地算出按时间比例的kn，这里k是一个不同的基于机器的恒定值。只有在开始之前安排好数字尺寸的绝对范围（64位通常已足够），这些算法才能够工作。

另外一个方法是产生一些不同的对系统的攻击，然后看这个系统是否足够强大以抵制这些攻击。这当然可以显示系统的强大性，但是同样它也不是具有结论性的。没有办法能够确保已经试过所有可能的攻击。虽然你认为你做得很彻底，但其实并非如此。

探索以语法为基础的仿真函数的局限性仍然是一个很重要的任务。现今存在的最好的理论领域是基于探索计算机的局限性并不断学习的基础上的。在这个领域里，很多研究者都把他们的工作基于Les Valient的PAC（Programmable Automatic Comparator，可编程自动比

较器）的概率论学说模型的基础上（参看[Val84]）。在这个模型里，计算机从一个特殊的层面上被给出几个例子，并且它必须试图了解有关该层的尽可能多的信息，以便它能够决定一个新的例子是否是该层的一个部分。对计算机是否成功的测量是大概性的，并且如果它得到的正确比错误更多的时候，它就成功了。

PAC算法还有很多不同的形式。在一些形式里，计算机给出几个在这个层面的例子。而在其他一些形式里，计算机在层内和层外都得到一些例子。有时，计算机甚至能够调制这一些例子并且询问这些例子是否在层内或层外。这个算法类型有最大的潜在能力，并且如果理论问题与其相悖时，它能提供一些帮助。

Michael Kearns和Les Valient在[KV89，Kea89]中显示了"学说"：布尔数学体系公式，有限时序机和恒定深度的门限电路（这至少和RSA加密的转换以及因子分解Blum整数（存在这样的x，使$x = pq$，p，q是质数，并且p，$q = 3 \mod 4$）一样难）。通过计算每一个不同计算模型的因子分解过程，就能把它们显示出来。

Dana Angluin和Michael Kharitonov[AK91]延续了Kearns和Valient的工作以及Moni Naor和M. Yung[NY89，NY90]的工作。这个工作显示了没有已知的算法能够在一个层里预测层次从属关系的多项式时间里运行，这个层是由有限时序机或与上下文无关语法的有限的合并或相交所定义。

这些范围可以预测一个句子是否在由一个语法定义的一个层次里，而不是发现它的语法分析树。但是，如果有某种方法能使一个语法分析树发现算法可被用来预测从属关系，那么，这些结果在这里就可以应用于以语法为基础的系统。

设想这样一个算法存在。以下是怎样把它应用于预测某个语言里的从属关系。我们设这个语言是由已知$L(G_1)$的语法G_1定义的。G_1的开始字符是S_1。现在，设有另外一个以S_2为开始字符的语法G_2。通过产生一个新的开始字符S，且S成分为$S \rightarrow S_1 \| S_2$，我们就产生了一个新的语法G，它是G_1和G_2合并的。取一组字符串$a_i \in L(G)$，它们既可在$L(G_1)$里也可在$L(G_2)$里。应用这个算法可以预测分析树，并且提供这组字符串。如果这样一个算法能够预测树语法（译者：tree grammar）的分析，那么它就能预测一个字符串是否在$L(G_1)$里。如果这样一个算法在多项式时间里运行，那么它就能被用来破坏RSA系统，用来因子分解Blum整数，以及解决其他的问题。因此，没有一种已知的算法能够预测到甚至是一棵语法分析树的第一分支。

把这个结果应用到现有的最难的语法里，对于怎样实际地产生一个语法，它不会提供任何线索。很需要找到一个算法，这个算法要可以构成这样一个语法并且使得该语法很难被发现。尽管有一些局部的观测，但它还是令人满意的。

可以很容易地设想一个能被很轻易地破坏的语法。如果每一个词或字符串在一个句子成分中是可见的，那么把产生一个很长字节的句子成分串隔离开相对来说是比较容易的。通过累积足够多的取样文字以便每一个句子成分都能被使用两次，建立句子成分的边界就变得很简单。这两个发生事件可以和揭示句子的不同部分相比较。

这导致了每一个单词都应该在复合的句子成分里出现的观测结果。在7.2.3节"加密语法"描述了扩写和缩写怎样被自动地应用改变语法以便它们能够适应这个需要。

要多少扩写和缩写才是足够的呢？有一本书[Way95]给出了可用来测量一个语法的"随机性"或"平均信息量"的一组方程式。这些方程式是以Shannon（译者：仙农，信息学家）

的一位流量的平均信息量测量为模型的。如果一个字很可能跟在另一个字的后面，那么在这个选择里就有充足的信息范围。

这些方程式可测量由一个语法产生的整个语言的平均信息量。如果这个数字很大，那么语法的信息容量也很大，而且在重要的循环出现之前，应该有大量的信息能被传输。这个实践性的方法可以很好地估计出一个语法的强大性。

这两个方法都显示了要发现产生一个文本的语法是很困难的，它们并不能保证安全性，但是它们却显示了在所有情况下，发现都是很难完成的。如果语法在扩写和缩写过程中被修改，那么发现将变得更加困难。通过同意一个加密安全伪随机数字生成器的同步方法，这些语法才可以被一个通道的双方所选择。

7.2.5 以仿真为基础的有效代码

本章描述的仿真系统有一个问题，即它是没有什么效果的。通过转化成文字的方式，即使是非常复杂的语法也很容易使一个文件的尺寸增加2倍、3倍或4倍，一些不太复杂的语法很容易产生10倍于输入的输出。这是完成一些看上去很不错的事情所必须付出的代价，但是如果它是安全的，那么这个算法可能还有其他的用处。

有效的加密算法当然可能使用这一章的技术。结果看上去像是普通的二进制数据，而不是我们说产生的文字，但是它们不会增加一个文件的尺寸。关键之处是建立一个大的语法。这里是一个例子：

- **终端**　设有256个终端字符，那就是说，一个字节的数值在0～255之间。我们称之为$\{t_0...t_{255}\}$。
- **变量**　设有n个变量，$\{v_0...v_n\}$。每一个变量都有256个成分。
- **句子成分**　每一个变量都有256个形式为$v_i \rightarrow t_j v_{a_1}...v_{a_k}$的句子成分。也就是说，每一个变量都将被转变成一个单独的终端值和变量k。一些句子将没有变量而一些则有很多变量。每一个终端都只在句子的右边出现为一个特殊的变量。这确保了语法分析的简单性。

当编码时，这个语法就不会增加文件的尺寸。每一个变量都有256个不同的成分可利用，这样在做选择的过程里就有8位被消耗。结果是一个新的加到流量上的终端值，并且它储存了8位。

这个系统也有一些潜在的问题。最大的一个问题是保证语言中的终端平均字符串是有限的。如果句子成分的右边有很多变量，那么产生过程就永远不会结束。处于等待状态的变量堆栈，其每一个成分都会继续不断地增长。解决方法是保证句子左边的变量平均数小于1。在变量平均数和由语法定义的语言里的短语和平均长度之间有一个直接的联系，那就是一个更小的变量平均数意味着更短的短语。随着变量平均数接近1，短语平均长度就趋于无穷大①。

这个语言里的一个短语的平均长度并没有在那个特殊例子里那么重要。因为这个语法是在GNF里并且没有必要保留语法分析的结果，所以这里的位可以很容易被恢复。每一个终端值只显示在每一个变量句子成分的右边，这样，最后的文件需要一个由这个语法产生的完整的短语，它可能只是一小部分，但是更复杂的语法则需要由开始字符产生的完整短语。

①这是以queuing theory（排队论，研究等待时间的等候理论）为模型的。

7.3 小结

本章描述了使用人为定义的系统规则来产生现实的文字的一些简单的方法。为什么复杂的语法不能在看起来如同胡言乱语的干扰中隐藏大量的信息，这是没有原因可解释的。这些胡言乱语可以发送到因特网的某个新闻组中，并且它很难辨别出这个胡言乱语与随机激情，以及通过语言传播的多层注释之间有何不同。

其他的抽象层次仍然是有可能的。MUD（Multiple-User Dungeons）允许用户在一个由原文设计者定义和建立的，以文字为基础的世界里相遇。相遇的人恰好在MUD房间里，并且用和你普通交谈时一样的方法进行交流，这种情况是有可能的。一些MUD现在已经可以运行那些假装是伟大的Eliza精神的人的计算机程序（参看[Wei76]），这些程序使用复杂的语法来指导计算机的响应。为什么由这个计算机做出的随机选择不能被转变以保存数据，这是没有原因可解释的。

这里有一个极端的例子。如果想跨越国家和一个朋友建立交流，通常，可以使用基本会话协议建立一个以文字为基础的链接，或者可以使用因特网电话程序实现声音的交换。在这两种情况中的任意一种，所交换的位都会受到监控。

> 太阳是一个贼，它用巨大的吸引力抢夺了辽阔的大海；月亮是一个臭名昭著的贼，它那皎洁的淡光是从太阳那里掠夺过来的；大海也是一个贼，它的波涛汹涌把月亮溶解成了带咸的泪水……
>
> ——摘自莎士比亚《在希腊的泰门里》。

如果你的会话程序并没有和其他人直接联系怎么办？作为代替，它将作为一个人记录进网上的某个MUD房间。另外一个人的会话程序也可以这样做并且进入同样的房间。由于空间的原因，我们设计一个有厚软垫的皮革椅的、充满烟雾的房间，使原文中的人物在里面迷路。墙壁上有些加有厚软垫的庞然大物来补充椅子。

作为把字位移交给其他人的角色的替代，会话程序将把它们编码成如同最后那晚的棒球比赛的一个叙述那样的无碍的东西。它足够聪明，以致于可以访问在线数据库并得到实际的记分卡以确保叙述的准确性。当另外的人做出响应时，他的会话程序将用一个相似的语法编码数据。真实的交谈是一些秘密的内容，但是对在线上窃听的任何人来说，它听起来只是像一场棒球比赛的叙述。

为什么交谈双方不能使用同样的语法，这是没有原因可解释的。这个协定使得交谈双方有可能保持一个连贯的交流。在一个人评论有关Joe Swatsem的击打后，另一个人则可能会说一些有关Swatsem的状况，这是因为同样的语法可以控制编后记的内容。

完整的系统是一个陈旧、别出心裁地用代码谈话的警匪片的程式化版本。一个匪徒说，“嗨，它将花费你10 000个香蕉。”其潜在意思是令人惊讶的。

- **伪装** 以语法为基础的仿真是非常现实的，惟一的限制是某个人产生这个语法所用时间量。
- **安全性** 处在最佳状态下，这个语法为基础的系统和RSA系统一样是很难被破坏的。虽然如此，并不意味着这个评估可以像RSA那样可以很容易地完成其安全性。没有

一个强大的模型能够产生一个好的密钥，也没有任何延伸的工作可以试图破坏系统。

- **怎样使用** 这个仿真系统的代码可以在附录A中找到，或者如果想让你的信息看上去像Spam，也可登录网站www.spammimic.com。
- **更进一步的工作** 在这个领域里还有很多方法可以采取。用一个具有更强能力的评估理论来辨认一个语言是很理想的。一个强而有效的判断，能在某个人弄清一个语言的意思之前估计出这个语言中有多少字符串必须被暴露，这是很理想的。如果某个人能编程估计出书[Way95]中的平均信息量或提出更好的平均信息量，那么我们就可以用它们做实验，并且可以看到它们如何有效地估计攻击的难度。

  有一个自动方法是很不错的，这个方法扫描文本并且产生这个系统使用的语法。被重复使用的语言中有很多基本的结构，如果有一些东西从网上的原料中提取，那么它就可以直接推进一个发送信息的程序，这真正导致了自动广播系统的产生。一部分将扫描新闻组或网络以便能得到产生语法的源文本，另一部分将使用它们把信息传播出去。设想它会导致一些奇怪的AI（Automatic Input，自动输入）技巧，它可以建立两个相互干扰，模仿网络的机器，但是这些机器却能真正地交换有价值的信息。

第8章 翻转与反向

Doggie的生活小片段

一个周末我开始了我的自动电唱机的制作，我想把它压缩在一起以便使声音听起来像小鸟的鸣叫，当导线通过电流，记录旋转回来时，我听到了我想要的愉快的声音：

Whoopee Tie Yi Yay,

世界变得更加美好，你的爱变得更加强大；

Whoopee Tie Yi Yay,

你的跛足的小狗将行走在这首歌的末尾。

这种音乐虽然有点奇异、异常和超现实主义，但是也没有恶魔。律音响铃非常美妙，就像水晶一样清晰告诉我们它都在一个水平上：

Whoopee Tie Yi Yay

这个周末你的60英尺游艇将漂浮在水上。

Whoopee Tie Yi Yay

老板会打电话告诉你，“你被升职了。”

所以片刻后我开始想：如果我重新导通接触音调，结果会怎样呢？在经过片刻的剪辑和接合后，电话突然响了，这是来自于电话里的声音：

Whoopee Tie Yi Yay

这是出版商的交换所，它告诉你你已经赢了

Whoopee Tie Yi Yay

一部新轿车，一大笔美元，和一栋在阳光下的房子。

几分钟后我失去的恋人就打电话给我说：和她一起私奔的小伙不是Jack。他带了假发，而且那卡车是他妈妈的。现在她乞求我让她回来。

Whoopee Tie Yi Yay

为什么把你的悲伤花在一个破灭的将来

Whoopee Tie Yi Yay

为什么当事后才知道如此完美时才往回看。

8.1 反向运行

由这首歌我们引入了本章，歌的内容是关于当一个男人在他的电唱机反向运转后发现了一个演奏乡村音乐的好方法时，所发生的一些事情：他跛脚的狗会走路了，他的女朋友回到了他身边，金钱也蜂拥而来。本章的目的是建立一个机器，使它在向前运行的时候能够隐藏数据，而反向运行机器则能恢复数据。使用这样一个机器的主要优点，是有一些理论校验证据显示，这个机器可以不受到计算机的攻击。这些关于系统如何强大的理论估计并不需要实际的、有目的的可靠性，但是它们阐述了一个令人感兴趣的潜能。

第7章描述了怎样使用语法在现实的声音文本里隐藏数据，该系统的强大性来源于语法的结构和它们从一个简单的输入集合里产生很多不同句子的能力。这个系统的弱点也是相当明显的，与上下文无关的语法不能真正地保持对球类比赛或其他更复杂的主题的跟踪记录，它们只产生不考虑上下文的句子。虽然一点点灵巧就要走很长一段路，但是每一个试图产生复杂语法的人都开始理解了这个模型的局限性。

本章集中讲述一个更坚固和完全的模型，那就是著名的“Turing machine”（图灵机，一种可不受储存容量限制的假想计算机）。它是以Alan Turing的名字来命名的，Alan Turing在20世纪30年代建立了一个模型，作为探索计算所受的一些限制的工具。虽然这个模型没有提供一个促进能更好地仿真的好方法，但是它提供了一个更深层次的理论的考虑，就是考虑到了破坏系统有多难这个方面。

理解与上下文无关的语法局限性，一个好方法就是检查需要辨认的机器类型。在附录A里建立的从仿真中恢复数据的剖析器，至少在理论上，也是以向“push-down automata”（向下推动的自动操作）而闻名的。“automata”是指一个简单的假定嵌套和转述机制。“push-down”是指可利用的储存器类型——在这种情况下，一个push-down堆栈可通过把信息推入数据堆栈中来隐藏信息，并且通过把它从数据堆栈中拉出来重新得到信息。很多人把这个和自助餐厅里的盘柜相比较，盘子储存在一个弹簧承载的堆栈里。这种类型的储存器的主要局限是顺序，信息位只能用与它们被推入堆栈顺序相反的顺序从堆栈中被调回。没有方法能探索到更深的地方。

提供一个固定的检验证据是有可能的，它可以证明“push-down automata”是描述与上下文无关的语法行为的理想计算模型，但是解决方法有点单调。一个更好的解决方法是用一个语法来阐述它：

> **开始** → 德尔玛和露丝 **做什么** **在什么时候** ‖ 哈里和露丝 **做什么** **在什么时候**
> **做什么** → 去射击 在什么地方 ‖ 购买保险 **在什么地方**
> **与谁一起** → 与鲍勃和雷蒙德 ‖ 与莱温妮和雪莉
> **在什么时候** → 星期一 ‖ 星期二 ‖ 星期三～星期四
> **在什么地方** → 美国堪萨斯州 ‖ 加拿大

你可以在[AHU83]里找到一个很好的校验证据。

由这个语法产生的一个典型句子可能是“德尔玛和露丝与鲍勃和雷蒙德周一去堪萨斯州射去”。这是对每一个变量组合在一起的第一选择，并且隐藏了6位000000。但是当作出

了第一选择，并且德尔玛和露丝变成了句子的主语时，关于日期的问题就需要被贮藏起来，直到后来需要它时才拿出来。既然可以把句子看做首先是从最左边的变量发展的，也可以把句子看做是从堆栈中最顶端的变量选择开始的。一个句子是怎样产生的如表8.1所示，它阐述了上述两种方法。

表8.1 一个反向的产生

堆栈	待选的句子	变量的选择
开始		**名词**
做什么	德尔玛和露丝	德尔玛和露丝 **做什么**
在什么时候		**在什么时候**
与谁一起	德尔玛和露丝去	德尔玛和露丝去
在什么地方	射击	射击 **与谁一起 在什么地方**
在什么时候		
在什么地方	德尔玛和露丝去	德尔玛和露丝去
在什么时候	与鲍勃和雷蒙德射击	与鲍勃和雷蒙德射击
		在什么地方 在什么时候
在什么时候	德尔玛和露丝与	德尔玛和露丝与
	鲍勃和雷蒙德去射击	鲍勃和雷蒙德去射击
	在堪萨斯州	在堪萨斯州 **在什么时候**
空的	德尔玛和露丝与	德尔玛和露丝与
	鲍勃和雷蒙德去射击	鲍勃和雷蒙德去射击
	在堪萨斯州于星期一	在堪萨斯州于星期一

两种比喻的结果相互都很接近，与文本无关的语法和基于堆栈的解释机器是等价的，这也说明了为什么模仿一场棒球比赛中像出局数和打击数这样的特定细节是有可能的，而如果不能给出一个分数的模仿，那么阐述它则是更困难的。没有什么方法能在堆栈上重新安排信息或把不按顺序的信息辨认出来。

图灵机的建立和一个计算机模型大概一样。不像“push-down automata”那样，图灵机能在任何时间访问它的储存器的任何部分。在大多数模型中，它被描述为一盒“磁带”，而磁带的读取是通过磁头从左到右的扫描来完成的。也可以把磁带看做是规则计算机储存器，这个储存器的第一个字节地址为0，第二个字节地址为1，如此等等。

使用图灵机的最主要优点是能够在任何时间访问它的储存器的任何部分，这样就可以用地址10140和10142在存储器字节存储棒球比赛的比分。无论什么时候需要它，你都可以把比分拷贝到输出端。这种方法并没有提供任何特别大的语法模型，从而使人们更加容易建立一个处理翻转仿真发生器。这真令人叹息。

探索图灵机的真正原因是：有大量的理论结果暗示了分析图灵机的方式是有局限的。Alan Turing发展图灵机原本是为了探索计算机能做什么和不能做什么的局限性[Tur36a, Tur36b]。他探索的最大结果是显示了当计算机变得对自己不利时，计算机是怎样无能为力的。计算机及其当前运行的程序很少能决定性地告诉我们另外一个计算机程序。

这些结果和Kurt Gödel的工作非常相似，Kurt Gödel原本是做关于逻辑系统的相似工作的。他的著名定理显示了所有逻辑系统都是既不完整也不一致的。因为人们对用不完整的

逻辑系统工作已经相当满意，所以，这个结论对数学本身基本上是没有什么影响的。但是，这些结果却腐蚀了现代人的信念，这个信念是：技术能使世界更加完美。

> 林肯是真正清晰明白地说出这个概念的第一个人。那时，他告诉整个世界，“你可以在所有的时间里愚弄某些人和某些时间愚弄所有的人。但是你却不能在所有的时间里愚弄所有的人。”如果用“计算机程序”或“图灵机”来代替“人”的话，这个概念同样是对的。

Turing发现同样的应用于Gödel的逻辑系统也能应用于依靠这个系统运行的计算机及计算机程序。他举例显示了没有任何计算机程序能够决定性地响应另外一个计算机程序是否完成。图灵机也能找到一些计算机程序的子集的正确响应，但是它不能找到所有计算机程序的正确响应。这个程序是既不完整的也不一致的。

其他人已经把Turing的结果进行了进一步的扩展，他们展示了响应机器以说出任何有关于计算机的起决定性的东西是尤其不可能的。Rice的定理显示了计算机只能响应关于其他计算机的一些微不足道的问题[HU79]，微不足道的问题被定义为那些一直是真的或一直是假的问题。

在某种程度上，这些结果只在一定的理论高度上引起人们的兴趣。毕竟，Macintosh（Apple公司于1984年推出的一种系列微机）计算机能检查一个为IBM个人计算机编写的计算机程序并且决定能不能执行它。在多数情况下，一个字处理器可以查找一个文件并指出它是否在错误的格式里。在多数情况下，连接因特网的计算机能够建立一个和其他因特网的计算机的连接，并且判明其他计算机是否是在使用正确的语言。为了很多实践性的目的，计算机能够完成我们指示它们做的大多数事情。

这里使用的限定词是“在多数情况下（most of the time）”，是因为每个人都知道软件可能是多么地不完美和多么地脆弱。由文字机引起的问题是非凡的，它们做它们被指令做的事情，并且通常是不完整或不完美的——就像理论模型预测的一样。

你能中断一个病毒软件吗？你能命名一个病毒吗？一个病毒会有多聪明呢？

我们面临的问题被事实复合了，这个事实是：这个应用并不像展开的字处理文件那样直接。目的是隐藏信息以便它不被找到。没有什么操作能揭穿信息保护器与攻击者之间的神秘的面纱。一个更好的模式是计算机病毒世界，在这里，一个人正在产生一个可以通向世界的计算机程序，而某些其他的人正在试图编写一个能够禁止一个病毒的反病毒程序。今天，标准病毒扫描程序已经建立，以寻找属于病毒的一部分的寄存器命令串。如果这些串被发现，那么病毒就肯定在那个地方。这种检测程序类型很容易编写并保持是最新的。每一次一个新的病毒被发现，一个新的寄存器串就被加到清单上。

但是更精明的病毒正在准备中。有很多相似的命令串是执行一个类似于病毒的工作，病毒可能选择任何由这些命令构成的组合。如果病毒用每一种模型给其本身加密，会发生什么呢？如果一个携带与上下文无关的命令，能产生有效的病毒，结果又会怎样呢？每次它都把自己拷贝进一个新的计算机或计算机程序，病毒使用语法作为其副本后，就会生成一个新的版本模型。检测一个像这样的病毒是一件更加困难的事情。

因为每一个病毒版本的次序是不同的，所以不能只是单单按照次序来扫描命令。需要建立一个更一般化的病毒属性模型，并且在继续做此事之前，必须知道病毒是怎样完成它自己的工作的。如果获得一个携带病毒的与上下文无关语法的完备副本，就可以产生一个语法分析来剖析每一个文件并且寻找来自于这个语法的一些东西。如果这些东西被找到，那么一个病毒就被确认了。这个工作会不时地进行，但是如果一个病毒修改了语法，而且使用的方法就像7.2.3节里那样：使语法扩张或收缩。那么会产生什么样的结果呢？可能有无穷多个可能的结果。

本章的目的是捕获相同的理论上的不可能性，这个不可能性赋予了图灵机抵制攻击的能力，方法是通过产生一个不止有一个密码的密码系统。它是一个既能向前运行也能向后运行的计算型的机器，如果这个机器有图灵机那么强大，那么至少有在理论上信息永远不会被揭露的可能性。建立另外一个计算机，使它能通过反向来攻击所有可能的机器。这种计算机并不是在所有情况下都能工作的。

8.1.1　换向传动装置

很多计算机科学家在相当长的一段时间里一直在研究可逆式的计算机，但并不是为了隐藏信息这个目的。可逆式的机器有一个热力学环孔，这暗示了这些机器可能会随着CPU变得越来越强大而变得很有用途。普通电子线路在每做一个决定的时候都浪费了一些能量，但是可逆式的机器就不会。浪费的能量作为热量留在常规芯片中，这就是为什么那些最新和最快的CPU都把热量操作散热片放到顶端。一些最快的机器是通过液体冷却剂来冷却的，这些冷却剂甚至可以吸走更多的热量。消耗热量的增长是一个很严重的问题——如果这个问题不被解决的话，CPU就会失效。

关于可逆式计算机最初的工作是理论性的和假设的。Ed Fredkin[Fre82]提供了一种不会消耗能量的逻辑门类型。普通的使用2位的与门是不可逆的。例如，如果x *AND* y是1，那么x和y都能被恢复，因为它们两个里面都肯定含有1。但是如果x *AND* y是0，那么就不能具体知道x和y，x和y两者可能只有一个是1。这使得反向运行这样一个普通与门是不可能的。

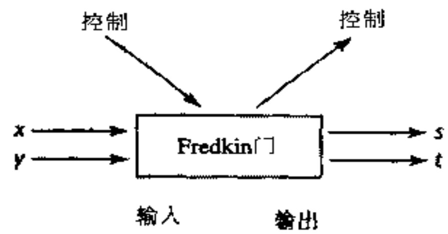
在另一方面，Fredkin门不会删除信息，所以它是可逆的。图8.1显示了这样一个门和传动这个门的一个逻辑表。图中有三条线输入，三条线输出。有一根输入线是控制线。如果控制线被接通，那么另两条线就被交换。如果控制线关闭，那么另两条线就反向。因为每一个输出只对应一个可能的输入，所以这个门是可以逆向运行的。

> 由Charles Bennett和Rolf Landauer所做的《科学美国论》为逆向机器提供了一个很好的介绍[BL85]。

图8.2显示了由一个Fredkin门建立的与门。来自于普通与门的两条输入线的一条被用做控制线。只有一条输出线需要给我们答案。另外两位则被浪费了。通常，通过把位发送到使芯片加热的接地点，这里的信息就被丢弃。真正的可逆机器就是在这个位置储存位直到计算机被逆向运行。一个或门也是用同样的方法建立的，但是它将有一个输入线被设为1。

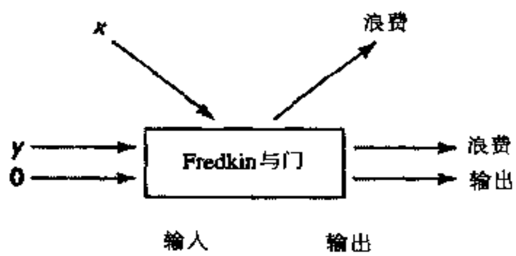
还有很多其他的机械方法可以建立一个可逆计算机。Ed Fredlom和Tommaso Toffoli发展了一个台球记分计算机，如果可以找到一个合适的台，那么它就可以逆向运行[FT82]。这个台必须相当地平滑以便使球都能够同步移动。台本身必须是无摩擦的，并且缓冲器必须把

所有的能量都返回给球以便没有能量损失，另外，在计算的开头和结尾都必须给予一个很大的动量。



控制	x	y	控制	s	t
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	1

图8.1 一个Fredkin门的阐述。如果控制线接通，那么输出线就被交换。否则，它们就独自输出[BL85]



x	y		浪费	浪费	输出
0	0	0	0	0	0
0	1	0	0	1	0
1	0	0	1	0	0
1	1	0	1	0	1

图8.2 一个由Fredkin门建立的与门。额外浪费的位必须在门里被存储以便以后计算可以逆向[BL85]

图8.3显示了两个台球是怎样建立一个门的。出现一个球表示处于接通状态。所以如果两个球都出现，它们就会相互弹回。只有一个球时它才能继续走它的通道。如果所有的球都到达了计算的末尾，那么就会被终点的一堵墙弹回并反向而行。应该很容易看出，这个门既能向前工作也能向后工作。或门则更复杂一些，并且，它还包括额外的操作球的墙。

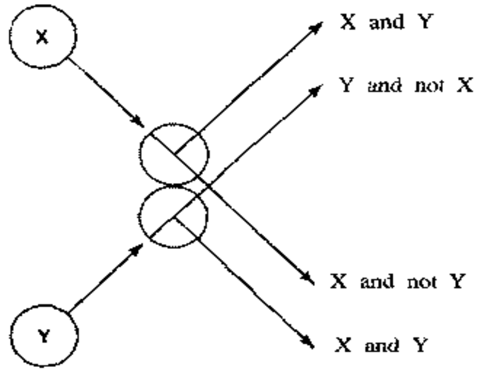


图8.3 一个台球与门的三种可能的结果。一个台球的出现表示处于接通状态信号。如果只有一个台球在场，那么就不会发生弹撞并且这个球会沿道前进。如果有两个球在场，那么就会相互弹撞。如果没有球在场，那么什么都不会发生[BL85]

这是一个令人感兴趣的概念，但是它很难有用处。没有人能建立这样一个无摩擦的材料。如果有的话，也只能是在我们实际使用它来计算之前的若干年（还有很多其他令人感兴趣的东西，像观看人们在上面打曲棍球）。尽管如此，更多的实践的工具还是使用，比如在它前后出现的使用细胞自动机。Toffoli在他的博士论文[Tof77a]及其他相应的文章[Tof77b，TM87]里描述了逆向细胞自动机；N．Margolus在[Mar84]里提供了执行台球模型的解决方法。

逆向计算机的关键结果应归功于Charles Bennett，他显示了用一个可逆图灵机可以处理任何计算[BL85]。为了移动磁带的读/写磁头和改变机器的状态，Charles Bennett用已定义好的命令产生了一些可逆图灵机的基本例子。这些机器的转换规则看上去和Fredkin门很相似，和每一步都会传出大量信息一样，每一步也会传入大量信息。这被准确地平衡，以便能推断出从一个位置通往另一个位置，并且机器能够逆向。

David Hillman写了一本书[Hil91]，*内容是关于可逆的一维空间网状自动控制。*

这个结果显示了任何一个计算机能处理的事情都能用一个逆向计算机处理。所有需要做的事就是从每一步中找到一种保存信息的方法，以使它能够有效地逆向运行。但是，如果想隐藏信息的话那将意味着什么呢？它将意味着任何计算在最后结果中都可以被用来隐藏信息。可以存储多少信息呢？这全取决于计算了。通常，任何用来增添现实性或给一场比赛结果加密的随机数字生成器，都能被一个将要隐藏的数据集合所替代。当机器逆向运行的时候，这个数据就可以被恢复。

可逆计算对排错程序也很有用处。

这样一个系统是怎样工作的呢？一个明显的解决方法就是产生一个通用的可逆图灵机格式。在个人计算机中运行的一个标准程序能够把数据读入图灵机并且使它向前或向后运行。如果想发送一个信息，就会把数据打包并运行这个机器直到它停止。运行结果就是输出，也许一些由计算机生成的诗或一堆没用的数据为了使机器逆向运行而必须保持不被丢弃。

在另一端，接收者将把信息装载进同样通用的可逆图灵机并且把机器逆向运行以恢复数据。这个方案有一个问题，就是任何攻击者都有同样通用的可逆图灵机。他们可以在中途截取信息并且使它反向。为了保证这种技术的成功，一些数据对攻击者来说就必须保持是秘密的，这可以单独地进行。在第7章里的语法机里，语法扮演的角色是密钥，它必须被单独地分配。

一个解决方法是保持图灵机（即，这是个程序）的结构是秘密的，并且把它看成是一个密钥。只有输出的和额外的"废"位信息必须传送给接受者。任何人都可以在中途截取信息，但是如果没有产生信息的一个程序拷贝的话，他们就不能阅读信息。

这有多难呢？很显然，有一些程序是很容易损坏的。例如，一个复制和吐出隐藏信息的程序很容易推断。输出将保持所有的原始文件结构，越来越多的复杂程序推导起来也将越来越复杂。最终，必须有一些是很难损坏的。一个艰难的问题是：是否存在某个阈值，它是在确定已知程序在阈值限以外的、完全安全的地方建立的。

这样一个阈值不能很明确地被定义。那就是说，没有什么机器可以检查任何程序并且说"这是不会被破坏的。"可能有些机器能够指出程序里的缺陷并且显示它们是怎样被破坏的，但是这些机器不能保证可以找出所有的程序缺陷。

这个不确定性是一个很令人苦恼的东西，但是它同样也影响了敌人。敌人不可能做出一个任意的机器，这个机器能检查你发送的每一个信息并且发现用来隐藏数据的程序。它也许能够找到一些解决方法，但前提是在所有情况下都不能有残忍的强制性攻击。

这是一个很好的安全性开端，但是它并不是绝对的，一次装填提供了一个相似的安全覆盖，只要密钥位是完全随机的话，就没有残忍的强制性攻击能破坏系统。什么是完全随机？在实践中，这意味着攻击者不能建立一个预测位模型的机器。实质上完成一次装填比可逆图灵机更容易，有很多完全随机信息源和噪声可以被用来当做一个一次装填的基础。在写本书的时候，很多安装有内置电视调谐器的苹果计算机出现，一个空频道的随机雪花干扰可以为这样一个补丁做一个很好的开端。

本书的剩余部分将集中讨论实际建立一个能用来在文本里产生隐藏信息的可逆图灵机。这一部分是基于第7章中的语法方法，因为文本是处理过程的一个很好的结果。没有理由可以解释为什么这个工作不能适应产生其他的仿真。

8.2 可逆转机器的建立

如果每一个图灵机都能以可逆的方法重新建立，那么每一个可能的机器都是转变成隐藏信息的工具的一个选择。显然，一些机器比其他机器更有趣。例如，贷款银行会用一些计算机程序来评估申请者的信用额度，这些程序会以"有资格的"或"没有资格的"来回应。那仅仅是一小位的信息，并且看起来好像任何人都不可能在那个位里隐藏信息。在另一方面，产生像Doom（著名的3D动作游戏）那样的复杂游戏世界的程序则会吐出数以十亿的位。在噪声里有充足的空间。设想如果某个秘密信息在一个攻击对象的摇摆中被编码，你可以通过加入一个因特网Doom游戏组来获得信号。信息可能会碰到伪装成指令的电信号，在那里它可以把攻击者在荧屏上绘制出来，你的Doom版本就可以抽取它。

可逆机器可以抵制错误。Peter Neumann讨论了怎样插入一个同步标记到文本流量里以使中止错误的继续传播[Neu64]。

本章将展示怎样建立两个不同的可逆机器。第一个是一个简单的可逆图灵机，它是作为热身运动而提供的，它以Charles Bennett的著作为基础，并且它还展示了怎样制定一个图灵机的标准特征，以及怎样调谐它们以便有（而且也只有）一种状态能够通向另一个状态。这使得逆向是完全有可能的。

第二个机器是一个第7章中基于语法的仿真机的延伸。那个系统只使用与上下文无关的语法。其意图是使得模仿任意的计算变得可能，而模仿计算的目的是为了增加由系统产生的文本领域的范围。通过分析，在这个机器里，数据被系统隐藏，但系统却不能重新恢复，除非通过逆向地运行机器。这意味着需要建立一个实践性的方法，以便沿途传输额外的变量和“无用的”位。

8.2.1　可逆图灵机

一个普通的可逆图灵机是由一组状态S，一组能在磁带中出现的符号Σ，和一组转变规则δ组成的。其中δ是告诉机器什么时候发生了什么事情。例如，如果机器在状态2，那么就可以确定δ，并且如果在读/写磁头下面的磁带上有一个记号σ_4，那么读/写磁头就应该在磁带上写记号σ_5，把磁头移到正确的写保护缺口，并且变换到状态s_{42}。这就是怎样编程一个图灵机的过程。这个抽象概念没有加工过，但是它使得跟踪的所有可能性变得更加简单。

把这样一个机器转变成反向运行是很直接的。主要的问题是寻找状态的组合和导致同样状态的磁带记号，也就是说，当机器逆向运行不可能的时候，那是因为有两个不同的先前状态导致了一个当前的状态。最容易的解决方法是保持不断地分离这些状态，直到没有混淆为止。

对每一个状态$s_i \in S$，建立一个状态、磁带记号以及导致状态s_i的方位（s_j，σ_k，L）的三元表格。就是说，如果机器在状态s_j，此时读/写磁头上记号为σ_i，那么磁头将写σ_k并且向左移动一步。这意味着如果机器反向运行并且发现它在状态s_i，其右边用记号σ_k表示，那么它就可以向右移动和改变到状态s_j，并且重新用记号σ_i写σ_k而不妨碍程序的运行。这就是说，这是一个正确的移动。

如果三元形式（s_*，σ_*，L）和（s_*，σ_*，R）在同一组里，那么就会发生冲突（s_*代表S的任意一个元素s_t）。这是因为机器很可能在某些地方结束运行，那些地方是指一个记号在左边和一个记号在右边的地方。你可以因为程序结构的原因来争论，这样一个组合可能根本就不存在，但是这些通常是很难证明的。

如果这样一个冲突发生，那么就应该创立一个新的状态并且把作用分离成多个部分。所有向左移动到旧状态s_i的三元坐标仍然指向状态s_i。尽管如此，移到右边的三元坐标将被移动指向新的状态s_j。s_j的转变规则将是s_i的复制。

在很大程度上，分离这些状态与找到一个存放“无用的”位的地方同样简单。**Fredkin**与门产生了一些必须储存的无用位。每分离一个状态就产生一个无用位。

如果有两个三元坐标形式（s_a，σ_k，L）和（s_b，σ_k，L），那么同样的分离过程必须执行。s_a和s_b这两个状态都导致了同样的记号，这个记号存在于读/写磁头当前位置的右边。进

行选择是不可能的。每重复一次，一个新的状态就被添加，转变规则就被复制和分离一次。

图灵机随着逆向而增长实质上应该是很显然的，在很多情况下，这可能是按指数增长的。为什么那么多人都想用这个方法来编程，这是没有原因可解释的。虽然这个例子很复杂，但它是一个很好的开始。

8.2.2 可逆语法生成器

本书的目的是产生这样一些东西，这些东西看起来好像是无害的，但是却在普通视觉中隐藏了大量的信息。第7章用一个与文本无关的语法很好地完成了这份工作，但是那种方法有很多局限性。本书的这部分建立了一个等价于可逆的图灵机，它能够做所有的基本计算，但它仍然是可逆的。通过模仿Fredkin门，它得到了很多的性能。Fredkin门仅是更换信息以代替损坏的信息。

在这个机器的设计中需要面对很多问题。这里列出其中的一些：

额外状态 在计算的末尾，将有充足的额外“无用”位留在附近，这些位都必须传送给接受者以便他们能够逆向运行他们的机器。

有两种可能的解决方法。第一种是通过一个不同的通道发送额外的状态，它可能隐藏在一张照片的最小有效位里，或者通过其他一些传送通道发送。第二种是使用一个机器的残损的版本把它编码为文本而不用修改任何状态。也就是说，减少机器的容量，直到它像第9章描述的与文本无关的语法机器那样。

较弱的可编程性 任何使用机器的人都必须提出一个可模仿一些文本形式的语法集合，创建这些东西是非常复杂的。而如果语言足够灵活到可处理很多结构的话，它就是很理想的。

这个解决方法是模仿第7章中语法结构。将会有很多变量和产品，但是可以使用可逆代码在发送途中改变产品。

使额外状态最小化 任何额外位都必须通过一个单独的通道在末尾传输，它们应该保持在最小值。

因为这个原因，所有的串都应该事先被定义为连续的，它们不能被改变，如果它们被改变了，那么所有串的最终状态都必须传送给接受者，并且这将是一个大冗余。

算术算法 算术算法一般来说是不可逆的。3 + 2是不可逆的。但是如果方程式的一半留在手头的话，它就是可逆的。所以添加寄存器A_1的内容，添加寄存器A_2的内容和把结果放在寄存器A_1里都是可逆的。A_2的内容可以从A_1中抽出以恢复A_1的原始数值。

在大多数的部分里，如果它们用这种格式被表达，那么加、减、乘、除法都是可逆的。惟一的问题是被0乘的乘法，这必须被禁止。

存储器结构 存储器将采取什么样的形式？很显然，普通计算机允许程序员一次性摄取和变换拥塞块。因为它需要很多的额外无用位在附近被储存，所以这是不可行的。拥塞块移动数据是不可逆的，信息的交换也是不可逆的。

因为这个原因，会有一个寄存器数组，数组里的每一个寄存器都有一个在某些情况下可被四舍五人的数字。寄存器的最后状态将作为额外状态传送到接受者，所以任何程序员都应该有目的地节省地使用它们。不幸的是，逆向的规则使得这变得很困难。

条件语言　大多数在一个程序的分支之间选择的条件语言都可以逆向，但是有时候它们也不能逆向。考虑到这样的情况，“如果*x*小于100，那么给*x*加1。否则就给*p*加1。”如果正在逆向运算并且要得到*x*是100，那么应该选择哪一条通道呢？要从*p*中减去1或不从*p*中减1？两个情况中的任何一种都是有效的。

一种解决方法是在临时变量里执行每一个分支储存的结果。当遇到条件语言时，合适的选择会被交换进这个位置。

这种方法禁止程序被改变用来选择一个分支的变量的内容，这把很多标准程序语法都排除在外。下面是解决这个问题的一个方法：

```
if x<100 then{
     swap x, k;
     k=k+1}
else {
     p=p+1;
     swap x, k;}
swap x, k;
```

循环　循环对程序员来是说是一个便利的装置，但是当一个程序必须逆向运行的时候，它们则通常是一个模糊的干扰。一个简单的例子是while loop（当型循环），通常是编写用来找到C的一个串里的最后一个元素，即一个计数器顺着串移动直到终止字符，一个零，被找到为止。反向移动这个串可能是很容易的，但是知道在什么地方停止则是不可能的。

如果结构被很好地定义，那么一个循环的很多问题都能够被消除。只是为循环的结果提供一个测试条件是远远不够的，必须确定循环的因变量，它将运行循环的内容，直到因变量到达它的初始设定为止。

这个结构通常不够强大。请考虑以下这个循环程序：

```
i=1;
j=1;
While (i<2) do {
    j=j+.01;
    i=floor(j);}
```

Floor(*x*)函数的功能是找到小于或等于*x*的最大整数。这个函数在它停止之前将被执行100次。如果它被逆向执行，那么在*i*被设置成其初始设定为1之前，它只能通过循环程序两次。很明显*i*是循环的定义变量，但很明显*j*占用了一大部分。

有两种方法可以解决这个问题。第一种是警告程序员并且希望他们在使用代码发送信息之前注意到这个错误，这在他们的手中留下了一定的灵活性。

另外一个解决方法是多抑制一些循环的特性。它们不能被限制在确定一个计数器和映射函数的循环，这是没有理由的，这个计数器随着每一个反复操作而不断递增，映射函数则是应用于表格中的每一个元素的特殊函数。它们都是非常有用的，而且都很容易逆向而不发生冲突。

递归　递归在这里也是一个问题。如果程序是命令自己的，那么它们事实上是在建立一个循环，并且确认一个循环的初始位置可能很困难。例如，下面是一个循环程序例子的开

头部分：

```
procedure Bob;
   x=x+1;
   if x<100 then Bob;
end;
```

这是一个当型循环，它不可能逆向运行。当它开始运行时，也不可能知道x的初始值。

一种解决方法是禁止所有的递归。标准循环的建立要满足大多数目的，也就是说，理论上是有疑问的。很多理论上很难对付的程序都是起源于它们开始递归的能力。当这使得执行可逆程序变得更容易时，它将严重地减小它们的理论安全性。

另一种解决方法是在所有受影响的变量进入程序之前保存它们的拷贝。所以，当程序Bob开始运行时，可逆机将保存x的一个拷贝。这个版本不会被损坏，它将变成无用位的一部分，必须随着输出一起传送。

> Ralph Merkle也指出很多汇编码都是可逆的，并且预测在将来，聪明的编辑者将会重新安排结构以确保可逆性。一旦芯片被设计能节省可逆计算的能量，这就允许芯片运行冷却器[Mer93]。

最后，这个系统的代码和规则机器使用的汇编码非常地接近，惟一的不同之处就是这个系统没有完全的重写信息，那将使得系统不可逆。也许将来的机器可以实际地改变程序设计系统，以提高可逆性。如果可逆计算机能够证明它是将假设力度减小到一个可以接受的程度的最好的方法，那么这一天就一定会到来。

8.2.3 可逆语法机

虽然结构和机器代码非常相似，但是作者还是选择在LISP里产生可逆语法机（RGM）的执行系统。这种语言是产生一种新的语言和反复考虑它们的限制的一个最好的实验工具，它包括产生和修改表格的很多基本特征，另外还有很多内置的模式匹配函数。所有这些都使得产生一个可逆机相对来说比较容易，即使它不具有很多现代编译器的特征。

以下是这个系统的主要部分：

恒定的表格 被程序作为其语法的主要短语，将在表格（恒定表格）里被储存。它是一个正式的表格对，每一个对的第一个元素是一个像标记一样的称呼，它被用来作为一个短语的简写。第二个元素是包含恒定数据的一个字符串。这个恒定表格是必须分配给会话双方的初始程序的一部分。恒定值是不会改变的，所以不必随着由向前运行一个程序产生的无用位来传播它们，这样就节省了传播费用。恒定表格的主要目的是保持所有的短语都是沿着这个途径输出。这些是长时间的，并且实质上改变它们是没有原因可解释的。把它们定义为恒定值可以节省空间。恒定表格也包括像可变表格那样的任何数据。数据却是不会改变的。

可变量 在变量里储存的数据必须事先被定义，以便它们可持有初始值。这些初始值只是被实际分配得到一个变量的信息的时间。剩余代码必须通过交换指令来改变数值。这些变量储存在变量表格里，变量表格通常也是一个表格对。其中第一个元素是变量标记名，第二个元素是储存在变量里的数据。

这里有五种数据类型可以利用：表格、串、整数、浮点数和标记符。表格是由任意5个元素组成。虽然没有严格的类型检验，但是如果错误数据被加进指令，那么这些指令就会失败。例如，添加两个字符串是不明确的。

应该谨慎地选择变量。这些变量中的恒定值需要发送到输出端以便接受者能够使代码逆向。变量越多，"无用"代码部分就越大。

程序列表 在每一个步骤里都必须有某个代码被执行，这些是程序过程。为了更加简单，它们真正是由标记符确认的指令列表，并且储存在程序列表里。程序列表也是一个表格对。其中第一个元素是标记符确认的程序，第二个是指令列表。虽然在目前的执行程序中没有参数可对照，但是为什么它们不被加入，这是没有原因可解释的。

指令 指令是操作数据的基本组成部分，它们必须是单独可逆的。它包括基本算术算法、交换指令、if语句和循环工具。这些指令采用的是经典LISP命令形式。放在指令前面的前缀表示法在这种情况下并不令人苦恼，因为算术算法是可逆的。所以加法看上去就像这样：add first second。这个指令加上第一和第二并且把结果放在第一位。

还有三种其他的算术指令：sub，mul和div，它们分别代表减法、乘法和除法。惟一的限制就是你不能用一个数与零相乘，因为它是不可逆的，这会导出一个错误信息。

输出指令 有一个特殊的指令：chz，它使用被隐藏的位从一个列表中挑出一个输出，当这个指令被接受者逆向运行的时候，隐藏位就从选择里恢复。这个格式非常简单：（chz（tag...tag...tag））。这个函数建立了一个像第7章算法里的霍夫曼树，并使用位做选择。流行的版本并不包括选择加权平均值的能力，但是为什么这个特征在将来也不能添加，这是没有原因可解释的。

标记符能够指出一个变量和一个恒定值。在大多数情况下，它们能够指出作为恒定值储存的字符串，那是最有效的情况。在某些情况下，这些标记符将包含其他的标记符，在这种情况下，选择函数简单地估算出标记符，并且把通路继续延伸，直到它找到一个输出串为止。

由于实践的原因，程序员应该注意可逆性的问题。如果两个不同的标记符指向同一个字符串，那么就没有方法能使隐藏位被正确地恢复。这些东西在事先是不能检查出来的。虽然程序可以在不工作的时候检查出这个，但是运行的执行程序却不能做到。

代码分支 一个if语句可以用来发送不同分支的估算结果，其格式是（if（test if-branch else-branch））。这个程序估计test，如果它是真的，那么它就允许if-branch，否则，它就允许else-branch。

这个测试的格式和普通LISP非常相似。例如，如果a比b大，那么测试（gt a b）就回到真值。另外的判定函数是lt，le，ge，eq，它们分别代表小于、小于或等于、大于或等于，以及等于。

如果两个用来判定的变量改变了其中一个分支，那么当前运行的、查找错误的程序就可能被引入。它查找错误是通过把名字推到禁止列表上，并且是通过在每一步操作之前检查列表来完成的。

程序计数器和代码 这个机器像大多数其他的软件程序一样，在tag主函数里有一个主要的程序。这是第一个执行程序，而RGM则是结束程序。其他的程序当被遇到时就被执行，

并且一个堆栈被用来跟踪部分程序结束时的位置。

源代码可以在附录C中找到，那里有两个主函数：编码函数和解码函数。一个将选取一个数据文件并用语法编码一个信息，另一个将解码一个信息。

8.3 小结

设反向运行一个机器是产生最复杂的计算机生成仿真的一种方法。也可以建立双层语法或其他一些修改过的基于语法的系统。

- **伪装** 由这些可逆机产生的文字可以和一台计算机一样能做很多事。但是并不是那么顺利的，计算机在真正愚弄一个人之前还需要很长的一段路要走。静态文字仍然是很现实的。
- **怎样实现** 对这个系统安全性的探讨甚至比理解第7章使用的与文字无关的语法更加复杂。从理论上说，没有图灵机能够对可逆图灵机作出非平凡的陈述。在实践中，也许有可用的算法能把正在使用的信息模式集合起来。对可逆机来说，怎样产生相当安全的程序和怎样破坏特定子集的问题一样公开。
- **怎样使用** 在附录C里给出了LISP软件。它是在可利用的XLISP软件上运行的，并且它在整个因特网上有很多是免费的。
- **进一步的工作** LISP代码刚刚起步。如果已经读取过一个LISP解释程序，那么你就可以很容易地使用它。一个更好的版本提供了一个更宽的代码选项，这使得编程复杂的文字变得更加容易了。

一个更令人感兴趣的问题是怎样保证其安全性。有可能为了测试可逆机的力度而编程产生一个机制吗？这样一个测试机制能够得到保障吗？最终的机制可能并不存在，但是产生几个攻击模型则是有可能的。每一种攻击类型都有一个相应的尺度，它可以估计出文法抵制攻击的能力。任何模型和尺度的结合都是令人感兴趣的。

第9章 噪声中的生活

噪声中的男孩，Idaho（美国爱达荷州）

场景：车库中，有两个十多岁的少年在练习吉他。

少年A： 不，我不想这个曲谱像这样，“啪啦，哒拉，哒拉，哒拉，呀，嗡嗡，啾……”

少年B： “啪啦，哒拉，哒拉，哒拉，哒拉，呀，呀，嗡嗡，啾……”怎样？

少年A： 嗨，我们妥协一下，就“啪啦，哒拉，哒拉，呀，啦，砰，呀”？

少年B： 哦，我不知道更多了。

少年A： 有什么问题？

少年B： 我只是厌倦了用噪声来说话。

少年A： 嗨，我们承认。Fishback女士在英语课上教过我们，真正的艺术家要接受当今社会的挑战。我们必须通过我们对艺术热爱的力量来揭露它的错误。我们的努力必须走过社会的基础过程，就像发生一个里氏10.0级的地震一样。

少年B： 哦。就这些啊。她只不过是一个嬉皮笑脸的女人，那就是她的想法。

少年A： 来吧。加入到聚会里来。我们不得不像成人那样面对迄今在这个星球上还没有出现过的紧急情况。我们必须要使文化合适地生存下来，我们必须告诉年轻人和那些坐立不安的人真相。有很多范例可以突破。

少年B： 可以闭上你那使用科学语言的嘴吗？Hornbeam先生说过Thomas Kuhn并不打算呆到期末。另外，人们在哥白尼和伽利略打破很多范例之前都设法过上了很好的生活。悦耳的音调和协和的声音又如何呢？

少年A： 我们可以让我们的模糊噪声巡回为我们做许多事情，郊区居民整个生活就像在梦游一样。只有我们刺耳的音符才能把他们唤醒到我们的绿色草坪的嘈杂声中来。那是Fishback女士说的。

少年B： 停止吧。你所能做的就是像你父亲的轿车那样。你的父亲把你从梦游中唤醒并且强制你走路去购物商场，这样你喜欢吗？我可不想把Fishback女士说的看成是美妙的音乐。

少年A： 为什么不？她显然很理解由下定决心要产生一个梦幻青春期的一个团体文化所提供的罪恶霸权。我们不仅仅是人，因为我们会消费。

少年B： Nirvana，Pearl Jam，和其他主要的商标在我们的购物商场都是标价出售的。

少年A： 哇，也许我们抱着反文化比传统文化更热卖的希望，并正在被引导走向背叛？

少年B： 是的，你就是背叛者。

少年A： 真的。当Fishback女士在她所教的班级不断地获得好的评分时，她只是想让我们成为一个好的青年。当20世纪60年代考试被恢复时，这个革命就终止了。他们抽烟、酗酒、抗议，但是这样做都是为了音乐和寻找喜爱音乐的人。然

后，他们就结婚工作了。我们只是在做他们的后代想做的事情。他们是看我们像他们梦中所见到的孩子，才和我们做交易的。

少年B： 你正在走入一个死胡同。

少年A： 比以前普遍深入的迅速增长只是另外一个市场动态。不知不觉，我们正在使我们的反叛力量，选择了通过由一个决心要毁坏每一个年轻人潜在剧变的公司及机构所产生的市场道路。我们正在消耗由伪反叛精神建立的反文化力量，并以此代替的是用我们的热情重新建立世界。

少年B： 本来就是。

少年A： 所以我们要做什么？

少年B： 我这里有一个贝多芬的音乐，这是没有副本的。

少年A： 太好了。通过使古典音乐复兴，我们将可以颠覆使用智力财产窃取我们年轻的公司音乐，代替为了节省17.95美元的Nirvava回顾CD而在麦当劳公司里长时间地工作，我们将通过从作者的和公司的强制性规则中得到长时间的自由演奏音乐而真正地破坏其权利机构。

少年B： 在第9交响曲中一些很低的和弦……

9.1 在噪声中隐藏

噪声是我们人类的生活的一部分。数字化和数字化广告试图给这个世界一个印象，那就是数字电路是无噪声的、更好的，但是，这也是只对了一半而已。不用改变信息就可以不断重复地复制数字信号，这要归功于误码校正代码和设计良好的电路，但是这样做不能消除很多原始噪声。数字相片、数字音乐和数字电影都携带有原始产生留下的一定数量的噪声。当声音和图像被转换成位时，这些完成转换的电路往往会有一些不完美之处（有漏洞）。一位的电子噪声可以轻微地改变所有的位，并且没有方法可以恢复。噪声在我门生活中无处不在。

噪声也是一个机会。如果位是否正确并不是很重要，那么任何需要隐藏信息的人都会利用这些不确定因素。他们可以通过擅自占用的权利来要求他们自己的位。这可能是密码学中最流行的形式，并且具有最大的潜力。网络中漂浮着数以百万的图像，它们可以被用做修饰网站的窗口，任何人都可以在里面“劫持”位来携带他或她自己的信息。

这个规则是很简单的。数字化的图像和声音可以用在空间和/或时间里某一特殊瞬间编码集中度的数字来表示。一张数字化图像其实是一个数字矩阵，它代表了从一个特殊时间的特殊位置里发散出来的光的集中度。数字化声音是声频信号瞬间拍打麦克风而发出的电脉冲。

所有这些数字都是不精确的，拍摄图像的数字相机不是很完美，这是因为把光子转变成位的CCD（电荷耦合装置电荷耦合器件）的排列会受到物理随机结果的支配。为了这个装置在标准空间层次下可以足够敏感地工作，它们通常必须只响应少量的光子。这个世界的随机性使得有时候会有大量的光子出现，而有时候却只有少量的光子到达。在很长的运行中这会被衡消掉，但是CCD必须在每秒一个分式里产生一个图像。所以短时间的中断是偶尔会发生的，麦克风也要经历同样的过程。

发送信息所利用的噪声数量是真正交错排列的。很多数字化图像是通过分配给每一个像素32位而被储存的，有8位是用来编码每一个像素中红、绿、蓝的数量或青、紫、黄的数量，总共就是24位。如果每一种颜色中只有一个像素被分配以隐藏信息，那么这大概是文件的10%。一幅Kodak（柯达）CD照片图像是3027×2048像素，并且占用了18MB空间。这样可为隐藏信息留下了大概1.8MB。本书的文字小于0.5MB，所以在一个单独的快照里有足够的空间来隐藏信息，很多人都不想把18MB的储存空间用在一个单独的快照上。图像模型的稍不准确性在200KB到600KB之间浮动，并且仍然可献出10%的空间来隐藏数据。

但是，如果大约有10%的空间用来隐藏数据，这对图像的表面有多大的影响呢？每一个8位都储存了一个0～255之间的数。每一个8位组的最后一个位称为最小有效位，它的值是1。第一个位，最高有效位，如果也是一个1，它就会为最终数字贡献128。这意味着最小有效位在最终图像里最多可以改变一个像素集中度的0.5%～1%。如果有一个方法，使用它交换图像数据的10%将大约影响最终图像1%，那么这个方法就是一个很好的解决办法。

为什么不能有更多的数据被储存起来，这是没有原因可解释的。如果有两个最小有效位被用来隐藏信息，那么每一个像素都不能改变超过3个单位。但这是最终图像尺寸的25%，这是一个等待人们捕获和使用的巨大带宽。

9.1.1 有关噪声的问题

在一个图像或声音文件的最小有效位里，可利用的带宽量是很大的，但是它并不总是很容易利用。不幸的是，为了获得隐藏的空间，潜在的密码学家必须与希望产生压缩算法以消除额外空间的压缩算法设计师相斗争。

基本图像格式使用24位编码每一个像素的颜色，但是这个基本格式越来越少被使用了。像JPEG那样的压缩算法可以很好地完成这个工作，并且通常只使用每像素1位或2位，另外，它还能够在不使图像失真的情况下节省10因素。例如，大多数数字照相机，现在都使用内置的JPEG压缩芯片来节省空间，这样就允许人们拍摄更多的照片。虽然24位颜色稍微更精确一点，但是没有人愿意在它上面浪费空间。

对音乐也是如此。今天，MP3文件比那些要在每一个时间块记录集中度的文件更普通。像MP3那样的压缩算法可以节省10因素的原始数字化数据，更新的算法则可以节省的更多。如果要把CD集储存在计算机里，这将是一个很好的方法，但是如果想建立一个为隐写术所利用的通道，就最好不要使用这个方法。

顺便提一下，这个影响导致了一些密码学家在一个文件的大多数“感知地有效的”部分里隐藏信息。那就是说，他们想忽略噪声，并且在人类所能感知的部分里隐藏信息。噪声最终将通过一些压缩算法被提取和删除，但是，可感知到的重要部分仍然存在[CKLS96]。作为替代在微妙的集中度变化里隐藏信息，现在人们可以在一个人的鼻子或长头发的某一个位置里隐藏信息。

这一点非常好，但是对研究者来说，这将是一个更大的挑战，而且它是一个松散的设计规则。即使是在一个文件里利用噪声的简单机制不可能有多强大，但是它们仍然是值得探索的。本章的剩下部分就主要讨论噪声，后面的章节则会尝试一些更强大的解决方法。

9.1.2 好的噪声

一个实践性的问题是找到好的噪声。大多数图像和声音文件都包括了足够隐藏3%变化的自然噪声，但是这个噪声很少是纯的。图9.1显示了一张桌子上的一台计算机的照片被黑白扫描后的图像。图9.2只显示了最小有效位，图9.3显示了次小有效位。很显然，它们有一个高度随机的模式，它是由扫描仪的数字化电路里的噪声所引起的。它是随机的，但不是任意随机的。与它们相对比，图9.4显示了这张照片的最高有效位。

第17章讨论了一些照相机怎样不能提供足够好的噪声以屏蔽隐藏的位。

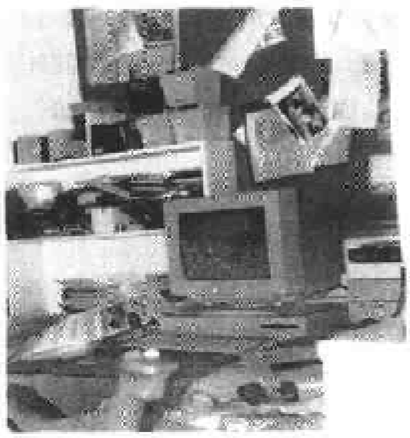


图9.1 这是一张作者的桌子的黑白照片。桌子上有很多隐藏秘密文件的杂乱的东西，图像中的噪声也可以为自己提供隐藏的数据

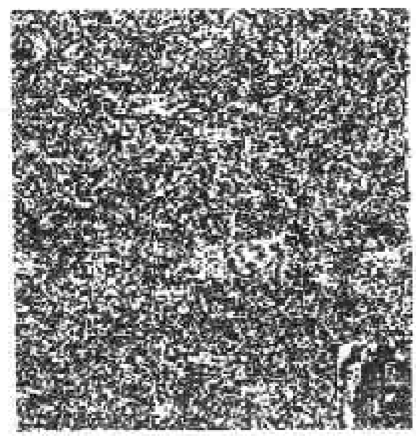


图9.2 这些是图9.1中照片的最小有效位。为了显示这个层上随机性的存在，最高有效位在这里被删除了（参看图9.4）

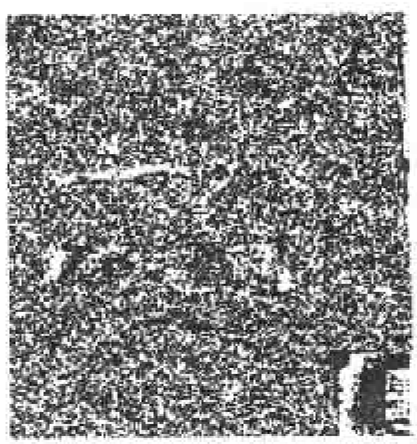


图9.3 即使是次小有效位也显现出很大地随机性。这些是图9.1中照片的次小有效位

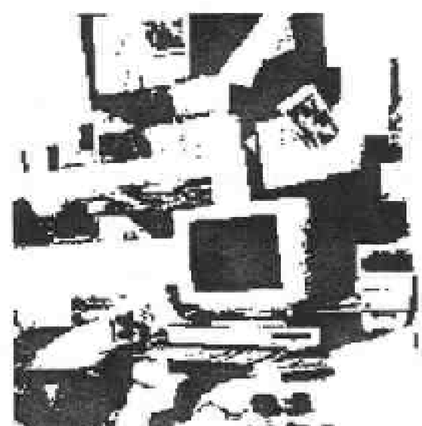


图9.4 图9.1中照片的最高有效位。为了与图9.2中图像形成对比，7个最小有效位被删除

很多图像和声音文件可能有足够的固有噪声来隐藏数据。图9.1图像就有足够多的杂乱碎片以致于小的变化不会显现出来，但是，有一些图像并不能很好地处理强加的噪声。很多图像完全是由Adobe（美国Adobe公司，著名的图形图像和排版软件的生产商）软件在计算机中产生出来的，它们产生了纯净的、一致的颜色区域。因为一个纯音调被转变成一个有一位噪声的音调，所以即使改变1位也会很醒目[①]。

9.2.7节“JPEG的使用”，显示了压缩是怎样确认一个图像里有多少空间可以被利用的。

9.1.3 独立性问题

有一个更深层的问题就是如何定义好的噪声。一个音乐或图像文件的最小有效位通常看起来接近随机，至少对于普通的眼睛和耳朵是这样，但是它们通常包含了隐藏的模型和结构。很多被用来产生文件的麦克风、照相机或扫描仪都有欠于完美，并且它们通常会引入它们自己的模式。

第17章提供了很多确认密码学存在的统计测试。

其中有一种最普通的模式是高顺序位和最小有效位之间的相关物。一张大白天的照片包括很多太阳的强烈反射，这些纯白点通常会使传感器饱和并且产生最大修补值255。还有254个相对较小的数值。如果最小有效位真正是随机的并且和高顺序位无关，那么255个数值和254个数值是等价的。这通常发生在图像其他部分的较小等级里，而且，这个图像里只有一种颜色并且这种颜色使图像饱和。

这是一个很简单的例子。传感器背后的那些具有渊博物理知识的优秀科学家通常可以发明出更完善的模型。例如，数字打印机的制造商可以调谐他们的调色剂机制以便使他们生产出来的打印机既能用全白线也能用全黑线工作。两样都要做好是很难的。

当用信息代替最小有效位时，所有这些微妙的统计模型都可能被毁坏。简单地插入一个被压缩和加密的信息会放进与高顺序位无关的白噪声。

要避免这个问题并不是很容易的。第6章里的仿真函数可以通过复杂的途径来模仿模型，但是这在很大程度上就像一个猫抓老鼠、间谍对间谍的游戏。攻击者可能有文件统计相关的某个模型，如果能预料到这个模型或者能提出自己的方法来对它解压缩，那么就可以在信息里给数据取模以适应它。如果你的选择不正确或者偷听者/攻击者改变了他的模型，那么数据就会像一个肿痛的大拇指那样伸出。

没有什么办法能够赢得这场游戏，但是使玩游戏的危险性最小化是有可能的。对这个统计问题的最好防御就是通过把数据打包分组来避免过于贪婪。一个800KB的文件有800KB的最小有效位可利用，它们能储存100KB的一个信息。尽管使用所有100个通道将彻底地毁坏所有的统计相关，但是插入一个1KB的信息将会保留99%的最小有效位不被改变。大多数统计模型也是不被改变的，这样就不能从一个纯文件中区分开来。

①纯的颜色通常对眼睛很刺激，这就是为什么艺术家使用（木材、岩石等的）纹理和稍微不纯的颜色来使得图像更有吸引力。任何人都可以猜测出为什么视神经会这样反应，但也许它是一种模式，该模式在一个太粗糙的水平里被数字化时，产生了一个像波纹干扰模式的效果。如果在电视上看到过一个人穿着一件格子花纹的裙子或戴着一条格子花纹的领带，就可以看到这个效果。

当密码学家变得贪婪并且使所有的通道都饱和时，所有对密码学的分析性攻击都会非常有效，保留大多数图像不被改变是最好的防御方法。

9.14 文件格式失败

文件格式对任何例行公事地使用位层加密术隐藏信息的人来说是一个严重的问题。很多图像或声音文件格式，都会采取挤出一些额外的噪声细节以便节省空间的设计。这可以通过一个像GIF（可交换的图像文件格式）那样的专门的、有效的文件格式来完成，也可以通过一个侵略性的压缩程序来完成，这个程序在重新建立一个不相同的图像情况下可以不用考虑。这两种格式都使得在一个图像的最小有效位里简单地隐藏信息变得很难。

是的，报纸说的是对的：在整个爱尔兰，雪是很普通的。——James Joyce，*The Dead*。

GIF文件格式和它的8位颜色标准是一个很大的阻碍，这是因为8位颜色和24位颜色是很明显的不同。它使用了一个具有256种不同颜色的表格把图像表示成一个颜色图，每一个像素的颜色通过给定与这个表格最接近的颜色而被描述。这些位并不与每一个像素的颜色集中度相对应。这意味着改变最小有效位并不必改变少于1%的单个像素集中度。表格的128项可能是一个饱和的宝石红，而129项可能是一个苍白，褪色的靛青。这些颜色只是在最小有效位上不同，但已足够引起最终结果的重要变化。

这个问题有很多不同的解决方法。第一是使用表格中较小数（前面）的颜色，代替选择256种颜色来完成表示图像颜色的工作，这个软件也可以选择128种颜色然后再选择和原来128种颜色相似的另外128种颜色，它们甚至可以是相同的，但是那会引起太大的怀疑，这可以被人为地做些安排，以便使两种相似的颜色只在一个位里不同。表9.1是这样的一个表格的缩写例子。

表9.1 一个GIF调色板的前5项

序号	二进制编码	红色饱和度	绿色饱和度	蓝色饱和度
0	00000000	150	20	10
1	00000001	151	20	12
2	00000010	14	150	165
3	00000011	16	152	167
4	00000100	132	100	10
5	00000101	135	67	15
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

所以，如果想在一个像素里隐藏数值1，就可以在表格中找到与其最接近的颜色，然后，用被设为1的最小有效位来选择它的模型。如果最接近的颜色有红色（设为第15项），绿色（151项）和蓝色（167项），并且想隐藏位1，那么就要为那个像素选择颜色3。如果想隐藏一个0，那么就选择颜色2。

没有原因可以解释为什么最小有效位需要被用来作为分离对。它对任何位都能很好地分离。表9.2显示了怎样使用第3位来作混合对。

表9.2 一个图像调色板的前6项

序号	二进制编码	红色饱和度	绿色饱和度	蓝色饱和度
0	00000000	150	20	10
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
2	00000010	14	150	165
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
4	00000100	132	100	10
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
33	00100001	151	20	12
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
35	00100011	16	152	167
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
37	00100101	135	67	15

也没有原因可以解释为什么在每一个像素里只有一位被编码。为一个图像选择256或128种最佳的颜色的相同算法可以用来找到最接近的64种颜色，然后每像素有2位被分配到隐藏的信息里。很显然，这会导致一个分裂的图像，但是隐藏的信息通常能缓和分裂的数量。例如，设想我们在试图隐藏每像素的4位，这将使得对每一种实际指定的颜色会有4位的剩余，并且在表格中将只有16种真正不同的颜色。如果照片里是一个人，那么背景里的颜色中有很大机会可能分配到绿色，有一种颜色：棕色，可以在人的头发中找到，并且可能有两种颜色被安排到肤色上。同色而浓淡不同的皮肤看上去会非常假①。但是这两种色调每一种都可以隐藏4位的信息，这意味着这两种色调每一种都有16个替代，并且这16个替代都可以被随机地使用。这将增加一些数量的纹理构造，因此还减小了一些影响②。

对以图像和声音为基础的密码学的另一个阻碍是压缩函数。这些数据的数字表示法是如此简单，以致于很多压缩算法存在，并且为了更好地在网络中传输，这些算法把数据分组

①最近的工作暗示人眼察觉皮肤颜色比察觉大多数颜色更敏感。所以为了更好的表示它们，最好的算法把表格里更多的颜色都归结为皮肤颜色。人眼真实上并不太注意一个树下的绿色阴影。

②随机噪声用来量化，这看上去更加现实了。太多的离散层看上去是人工的（参看[Rob62]）。

成更小的文件。JPEG是一个压缩照片的流行的标准算法，MPEG是为了移动图像而被设计的一个相似的标准。

JPEG算法也可以参看9.2.7节，“JPEG的使用”。

因为这两种算法都是有损耗的压缩函数，所以对位层的信息分组来说，它们都是危险的。如果采用了一个文件，用JPEG压缩它，并且随后解压缩它，那么结果实际上将与原始值不同。它看上去很相似，但却不是相同的。这个结果和在像文字那样的其他的数据形式里使用的无损耗的压缩是非常不同的。那些函数都是逐字逐句地重新产生数据。有损耗的压缩函数能够得到更高的压缩率，这是因为它们对这些细节采取了漫不经心的态度，只要最后的结果看上去足够接近就行。JPEG算法本身是可以调节的。如果愿意忍受更大的不精确性，那么就可以得到更有效的压缩。如果增加压缩的有效性，那么像素就开始混入同样的大颜色块里。

9.1.5 否认本领

否认本领是在网页中的GIF图像隐藏信息的最大特征。如果正确地构造信息，那么就可以在很多不相关的位置把它展开。如果信息被发现的话，要准确地判断它从哪里来是不可能的。

设想有一些位，想把它们分散到整个世界里。可以在一个GIF文件里简单地把这些位隐藏起来，并且为了方便所有的下载，需要把它放到主页上。如果无意识的人发现了这些位，因为它是在网页上，这些人就知道信息的来源。

为了替代，可以使用第4章里的基本技术把信息分离成n个部分，当这n个部分被异或在一起时，它们就会揭露隐藏的信息。通常，可以随机产生$n-1$个文件并且计算最后一个文件以便使所有的文件都加在一起。但是当网络上有一个很大的随机源时，为什么随机使用文件会有干扰呢？可以简单地从网络中获取$n-1$个不同的GIF图像并使用它们。一幅图像可能是从白宫主页下载的无害的照片，另一幅可能是从某个编辑俱乐部网页上下载的无害的摘录。任何想恢复信息的人将得到所有这些网络中的GIF图像，从它们里面恢复出秘密位，并且把它们加起来以恢复隐藏的数据。

这个诡计的网络效果就是否认本领。没有人能保证他知道谁是隐藏秘密位的人。是白宫？是编辑部？没有人知道是谁把秘密位插人进文件。即使是恢复位的人也不可能知道谁发送这个信息。这太狡猾了。

6.2.2节中“规则仿真和图像”，描述了在最小有效位里匹配模式的完善方法。

有一些实践的问题和这个技术相联系。第一，必须保持文件形成的数据是秘密的。GIF实际上包含了最新的文件信息。HTTP协议通常不把信息传送到网络浏览器上，所以把持信息秘密是没有问题的，但是也可以通过重新安排机器里的记时器来伪造它。

第二，应该找出具有存储秘密位的正确结构的GIF文件，这将阻止某个人检查所有的文件并发现只有一个文件具有隐藏位的正确结构。就是说，这n个文件除了一个以外所有的文件都是8位颜色，这8位颜色是在充满256种不同颜色深浅的颜色表格里，不是8位颜色的那个文件就是隐藏位的文件。这意味着，它们既可以是8位颜色也可以是24位颜色；黑白灰度图

像是最好的。

比方说，如果想产生3个不同文件组，那么就可以在这个方法上加一些误码校正特征。当3组文件混合在一起时，那么隐藏位的3个模型就出现了。文件之间的任何不同可以通过选择考虑范围内的位值（这个位值在3个文件中的2个里都是正确的）来解决。

比第3章描述的那些误码校正方法更复杂的方法也可以成功地使用。例如，一个将被隐藏的文件可以用一个误码校正代码来编码，比方说，这个代码把每8位转变成一个能恢复错误的12位数据块。每一个12位数据块中的一位都可以放入12个独立的文件里，然后，这12个文件就隐藏在散布在网络上的12个GIF图像里。如果某个人因为网络失败的原因而不能恢复所有12个GIF图像，那么误码校正代码就允许信息被恢复。

一旦数据被打包分组在GIF文件或声音文件里，就会有很多其他的技术可以被用来隐藏信息。这些文件已经包含了很多漂浮在网络上的信息，并且这些信息的比例将会不断地增加。新的技术提供了更多的机会。苹果计算机正在装载Quicktime（Apple公司的数字多媒体技术）VR，它是仿造三维计算的一种方法。很多致力于三维游戏工作的程序设计师都知道复杂的构造是在虚拟现实里产生逼真效果的最重要的部分，所有这些都是隐藏信息的最佳地方。

9.2 位移动

开始实验在噪声里隐藏信息的最好的方法，也许是下载一个漂浮在网上的实验包。隐藏信息的最老的方法中有3个，它们是Seek，EzStego和S-Tools。它们每一个都阐述了一个不同的技术来克服文件格式的限制。

附录D提供了一些更卓越的密码包的一个列表。

9.2.1 隐藏和查找（捉迷藏）

"捉迷藏"4.1版是一个为PC设计的基本加密程序，它是由一个只有通过电子信箱地址shaggy@phantom.com才能确认的作者编写的。这个程序只能对8位色或320×480像素的黑白GIF文件工作。这种320×480像素是最老的GIF文件格式的标准尺寸。该程序是在加或减数据之前显示图像。通用的存档包含了两个额外的程序，它们可以使得在GIF文件里隐藏信息变得更加容易。

第一个程序gray.exe把彩色GIF图像转变成不会显示任何与8位色加密术有关的人工因素的灰度GIF图像。第二个程序reduce.exe把颜色表格从256种颜色收缩到128种颜色，然后复制这128种颜色以便使颜色表中相邻的项互为副本。如果这样做了，在最小有效位里隐藏信息就不会影响图像的外表了。

有很多不同的程序可以减小调色板的尺寸。其中最简单的如下：

1. 生成一个二维矩阵，这个矩阵包含了调色板里所有颜色对之间的距离。两个颜色，即（R_1, G_1, B_1）和（R_2, G_2, B_2）之间的距离是：

$$\delta((R_1, G_1, B_1), (R_2, G_2, B_2))=\sqrt{(R_1-R_2)^2+(G_1-G_2)^2+(B_1-B_2)^2}$$

2. 找到最佳的颜色然后删除。就是说，用最接近的邻近值找到这个颜色并消除它。
3. 重复以上操作，直到调色板得到它所期望的尺寸为止。

第17章描述了检测加密信息的几种技术。

不幸的是，这个技术为每一个扫描GIF图像以寻找隐藏的信息的人留下了一个很大的危险信号。一个只有128种不同颜色的8位色表，比一个有很多人工因素的损坏图像更容易被自动地察觉。一个是很容易被测量的没有显著效率的量，另一个是通常表现在人眼方面的主观问题。图9.5显示了当最小有效位被改变时，一个24位的照片变化有多小，而图9.6显示了一个8位照片改变得有多厉害。



图9.5 这幅图显示了怎样给一个24位颜色图像的最小有效位增加信息以使得影响变得非常小。这个图像在这本书的后面被转变成黑白色（军事测距系统基础的照片惯例）

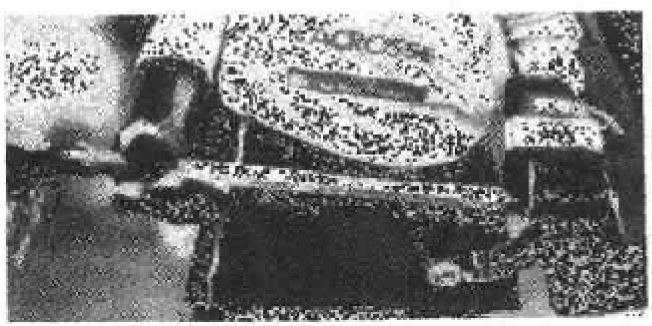


图9.6 8位的颜色图像可以使粗劣的选择物增加信息到最小有效位里，这是因为颜色表里的所有项都有可能相互相邻。这个图像在本书的后面被转变成黑白色

提供一个能确认不良人工因素的软件是可能的，但并不是很精确的。可以试图寻找颜色连贯性不大的地方，如果毗邻的像素颜色非常不同，那么就发出了一个信号，这就是一个由8位颜色产生的混乱了。一般来说，图像被数字化时会有少量的图像模糊和图形失真出现，沿着边缘毗邻的像素通常混合在一起，这些边缘是没有波纹的。即使是当边缘有波纹时，要测量颜色有剧烈变化的地方的随机性也是有可能的。如果这些呈现出是随机分布的，那么，它们就很有可能都来自于同一个过程。或者它们可能和Nirvana音乐是艺术等价的——设计低劣的图像用以推动完美的艺术，设计用以冲击和揭露生活中更丑恶一面的图像。有谁会知道呢？

源代码也包括了隐藏和查找（捉迷藏），这使得寻找程序的关键内容以检查出信息是怎样在位中扩散开来是有可能的。就是说，在GIF图像里有大约19 200字节空间可利用。如果一个更小的文件要被打包分组进入位，那么，这个程序就会尽量随机地安排这些位以便使得它们相互不相邻。这有两个结果。第一，噪声更随机地分布在图像中来代替被分组在图像的上半部分。第二，为了寻找数据，必须知道位的位置，并且这个位置是受由一个用户选择的密钥驱动的随机数字生成器控制的。这两个结果都增加了系统的安全性。

扩散度是由文件的一个8字节标题控制的，前两个字节是文件的长度，第二个两个字节是当信息被分组时随机选择的任意数字位，第三个字节对是版本号。第四个字节对不被使用，但是也被包括在IDEA密码器里，以填充完一个8字节块。这个块被IDEA密码器使用一个任意的密钥加密，并且被存储在图像的前64像素里。如果不知道这个密钥的话，那么就不能恢复控制数据在图像里的扩散度的标题信息。

实际的扩散是随机的。在一开始，随机数字生成器是由标题字组的第二个字节对查找出来的。"捉迷藏"4.1版里的代码简单地使用内置的随机数字生成器C，对大多数目的和意图来说，C已足够了。更强大的操作会使用一个加密安全随机数字生成器。或者，也许它会使用有一个专门密钥的IDEA来给随机位通量加密。这两种方法都可以增加很多的安全性。

以下这是致力于处理扩散问题的C代码的一部分。这个代码从颜色表项里获得颜色以形成像素（*x*, *y*），并且适当地交换和消除扩散变量数值，以控制平均有多少像素移动。它是这样被设置的：用输入数据的长度除以数量19 000，然后四舍五入。

```
int used=0,disp=0,extra=0;
for(i=0;i<8;i++) {
//Code removed here for flipping LSB of (x,y)
disp=(random(dispersion+extra)+1);
used+=disp;
extra=((dispersion*(i+1)-used);
x+=disp;
// Move over x pixels. Code removed to handle
//wraparound.
}
```

9.2.2 EzStego

EzStego是在一幅GIF图像的最小有效位里编码信息的一个基本程序，是用Java语言编写的。这是Romana Machado开发的第二个程序，他在早期也编写过了一个被称为Stego的程序。

在解决GIF调色板的问题上，EzStego方法是很简单的。这个软件把调色板进行分类，以使在一个n位文件里的2^n种颜色能够顺利平滑地从一个流到另一个。就是说，从第i种颜色到第$i+1$种颜色的每一步跳跃相对来讲是很小的。任何改变最小有效位的隐藏信息都会引入一些很小的变化。在调色板里给颜色排序对黑白图像里的一维颜色空间来说是微不足道的，但是在三维空间里这却难多了。最接近的颜色并不总是相邻的，图9.7显示了一个被分类过的调色板。

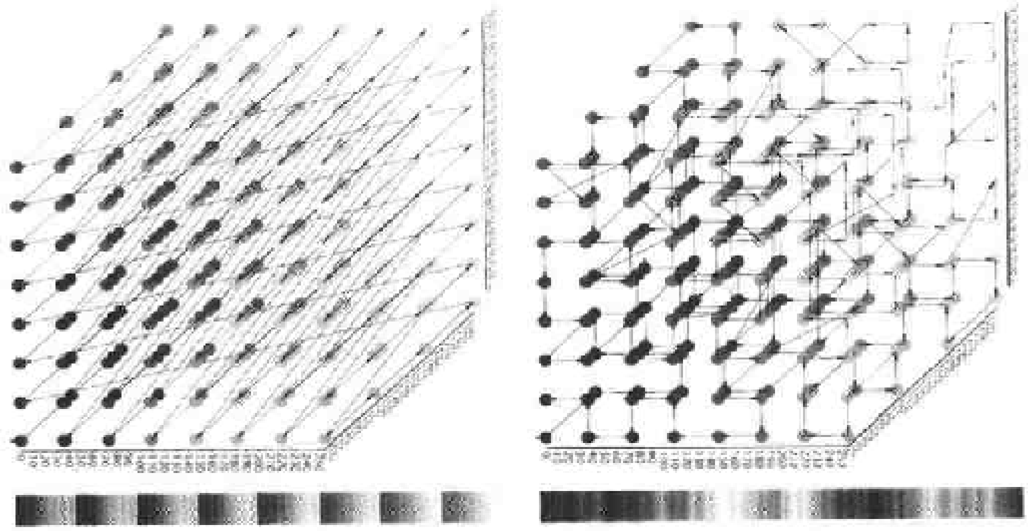


图9.7 左边的图显示了一个普通的调色板，右边的图显示了一个被分类过的调色板。一个颜色的数字模型10在调色板的对应颜色里插入显现出来（这些数字是由Andreas Westfeld和Andreas Pfitzmann在他们的著作[WP99]里提出的。他们只用了附录中的61条短线）

EzStego把这个问题看成是一个“货郎担”问题的模型。在三维空间里引用了3种颜色（RGB，红绿蓝），并且目的是找到一个最佳的解决方法，但是它通常能找到一个工作得很好的方法。这个算法如下：

其他的“货郎担”问题的近似解决方法可在书[Tah92，Cla83]里找到。

以表格里的两种颜色开始，$\{c_0, c_1\}$。设第一种颜色为C，重复步骤直到所有的颜色都被插入为止。

1. 找到离C最远的颜色，称之为d；
2. 找到在表格中插入这个颜色的最佳位置，就是说，扫描表格并且找到i，它要满足$\delta(C, c_i)+\delta(C, c_{i+1})$最小化；
3. 插入d并且设它为一个新的C。

找到最远的颜色保证了这个算法不会在三维颜色立方体的任何一个拐角“中断”，只是要使这种颜色到达另一种颜色需要做很大的跳跃。

这个分类算法并不是完美的，并且它不能保证跳跃足够小。当全部距离被最小化时，一些距离会比其他一些更重要。在最小有效位里加入信息只会产生一些变化。在一个3位的系统里，001只能改变到000，而不是010。对整个调色板进行分类会留下一个从颜色000到001的较长的跳跃，并且带进一个从001到010的较短的跳跃。一个更完善的方法是尽量找到使调色板里所有的颜色配对的途径，以便所有颜色对的总距离是最小的。

J. Fridrich对这个算法进行了一个简单的调整，替代给调色板分类，这个算法只是用奇偶校验寻找最接近的颜色以代表被隐藏的位。位的奇偶校验值是$R+G+B \mod 2$（参看[Fri99]）。

EzStego试图通过传输没有被分类过的调色板来阻碍密码分析。步骤如下：

1. 从一个由某个程序，例如Photoshop产生的未分类的调色板开始。
2. 对调色板进行分类以便最接近的颜色能够按顺序相互毗邻。
3. 通过移动最小有效位来编码信息，确保用一个加密安全随机数字通量来给信息加密。
4. 重新计算原始调色板里所有颜色的原始数值。
5. 传输信息，任何查看调色板的攻击者只能看到由无加密系统的程序产生的信息。
6. 接受者可以使用同样的分类算法对调色板进行重新分类，如果这个算法是确定的，那么它将产生相同的结果。
7. 接受者可以通过使用调色板里被分配到所有颜色的数值，实时地提取位。最小有效位也可以实时地编码信息。

因为这个算法使用了由图像生成软件产生的调色板，所以它可以解决复杂结构调色板的问题。任何攻击者都将很难通过查找调色板结构来确定信息的存在。

9.2.3 S-Tools工具包

在许多可利用的密码工具箱中，有一种叫S-Tools的工具包，它是由Andy Brown编写的。这个系统可以隐藏存储在GIF和BMP文件的图像里，存储在WAV（音频资源文件）文件的声音里，甚至存储在一个光盘中的定位信息段里。这个工具是在Windows 3.1系统下运行的，并且包括了很多帮助文件。这个程序本身是共享软件，支付15美元你就有权使用它，并且拥有一个源代码的副本。

Wilson MacGyver Liaw在[Lia95]里对GIF文件格式进行了一个全面的介绍。

在S-Tools的三个程序中，每一个都使用了一个加密算法宽数组提供给任何在一个文件里被分类的信息加密的最佳方法。这个软件允许你使用IDEA算法、DES、三重DES、MPJ2和NSEA。这些不同的算法中，每一个都能在5个最普通的反馈链接系统（CBC，ECB，CFB，OFB，PCBC）里的任何一个系统中使用。这显示了一个难以置信的完善水平。这些选项在商业系统中通常是不能利用的。S-Tools在每一个文件的开头加上了一个32位的随机位以确保这些链接方法可以安全地工作。

处理图像的S-Tools程序st-bmp.exe可以在GIF或BMP格式里阅读文件并且显示它们。数据被加到文件后，可以在图像处理前和图像处理后之间进行切换。这允许你对比结果并看到图像里是否有任何值得注意的变化。

有两种S-Tools支持的不同加密图像类型。第一种方法使用了24位图像并且对每一个像素里每3种颜色的最小有效位进行简单的改变。S-Tools的作者主张24位图像可能太浪费了，因为大多数人都不能使用它们——它们占用了太多的空间。这当然是正确的。当24位图像在艺术方面继续保持被普遍应用时，它们在网络上就显得没那么普及了，因为网络中占统治地位的是像GIF或PNG那样的8位格式。

这个软件的第二部分试图通过减小图像里颜色的数量来校正这个问题，所以图像中最后只有256种颜色。在大多数在视觉上可能不制造混乱的方法里，这个软件使用了由Paul Heckbert（参看[Hec82]）设计的算法来减小一个图像里的颜色数量。这个算法把所有颜色划

分在三维空间里，然后寻找包含所有颜色的一个n逻辑单元集合。当这个工作被完成时，它在每一个逻辑单元里选择了一种颜色来代表所有的颜色。S-Tools为如何选择这样一个颜色提供了3个不同的选项：中心颜色、平均颜色和所有像素的平均。

建立这组逻辑单元的过程在Heckberk的论文里被详细地描述。这个过程以一个完整的逻辑单元空间256×256开始，然后，它开始以尽可能最好的方法循环地对逻辑单元进行再划分，直到有n个逻辑单元代表空间时，这个划分过程才结束。Heckberk发展了这个算法来纠正那些正在被使用的“大众化”算法里的一些错误。这些算法会在颜色附近集成块直到这个块的数字达到n为止。然后它会选择某种颜色，通常是块中心的颜色，来代表所有的颜色。在紧密坚固的块里，这个工作可以做得很好，所有的颜色和被选择代表所有颜色的颜色之间非常不同。这导致了在细节中使用的颜色的一个大的转变。

Heckberk建议，理解这两种方法的一个好办法是把它们和“量化”方法相比较，这个“量化”方法在美国国会选择两个政府代表中被使用。参议院里每一个州都有两个成员，Heckberk就用这个和他的算法做比较。它展开表达式以便使颜色空间中没有任何部分是充分代表的或未被充分代表的。在另一方面，政府代表要能代表每一个人，如果你是从人口众多的地区，如曼哈顿岛（美国纽约）来的，那么这个工作就可以很好地完成。这些都代表了每一个城镇部分。然而，西部的州如内华达州，却只能有一个代表，这样它们在政府中就几乎没有权利。Heckberk把这个方法和“大众化”算法相比较。

David Charlap在[Cha95a，Cha5b]里很好地介绍了BMP格式。

S-Tools使用的微分算法能够使用两种不同的方法来分离逻辑单元。在一种方法里，通过测量RGB数值里最大的差异来选择最大的维数。在另一种方法里，则是通过对比不同选择的光度来找到最大维数的。这里详细地介绍了基本算法：

1. 把图像的所有颜色放在一个逻辑单元里。
2. 重复上个步骤直到有n个逻辑单元来代表最终的n种颜色。
   a. 对每一个逻辑单元，找到每一个维数里的最小和最大值。就是说，找到逻辑单元里任何颜色中红色的最小和最大值，找到逻辑单元里任何颜色中绿色的最小和最大值，找到逻辑单元里任何颜色中蓝色的最小和最大值。
   b. 对每一个逻辑单元的每一个维数，测量其长度。在绝对长度或光度里可能会有一些不同。
   c. 找到最长的维数并且分离含有此维数的特殊逻辑单元。Heckberk建议，这可以通过沿着维数找到逻辑单元里的中间位颜色来完成，也可以通过选择几何的中间位置来完成。
3. 选择一个具有代表性的颜色来代表每一个逻辑单元里所有的原始颜色。S-Tools提供了三种选择：逻辑单元中心、颜色平均值和像素平均值。

当这个新的n颜色组被选择时，S-Tools可以使用“高频振动”来用新的颜色代替旧的颜色。

通过一个中间过渡状态过程，这个算法可以找到新的颜色的最佳数字n。在数据被混合进入最小有效位后，它就慢慢地减小颜色数直到少于256种颜色为止。必须重复这个过程几

次，直到正确的数字被找到为止。因为S-Tools只是恒定地给每一个像素加3位，所以，它是不能预测颜色的最终数字的。就是说，它获得每一个像素的红、绿、蓝色数并且单独地改变每一种颜色的最小有效位。那意味着一种颜色很有可能会变成8种颜色。如果那种颜色在图像里是很普通的，那么这（一种颜色变成8种颜色）就很有可能发生，这是因为每一个像素都会以不同的方法处理。平均起来，8种稍微不同的颜色的每一种，都应该在10～12像素的相同颜色被绘制出来以后出现。

这意味着这个算法不可能预测出它所需要的颜色最终数。它试图把图像中颜色的最终数字减小到64。然后，在数据被混合进去后，它可能以270种或255种颜色结束。如果是255，那么它就可以保存文件。否则，它将重新开始这个程序并且减小更多的一些颜色。整个过程是重复的。S-Tools通过推断来预测正确的数字，但是每次修改文件的时候它都已经发生了几次重复过程。

由Henry Haster发明的MandelSteg程序，在Mandelbrot组的一幅图像的最小有效位里隐藏信息。这幅人工图像被计算成7位的精确性，然后信息就在第8位里隐藏。参看15.2.2节。

9.2.4 S-Tools和声音文件

S-Tools也包括一个被设计在WAV（声音资源文件）文件的最小有效位里储存数据的程序——美国微软公司的Windows操作系统标准声音格式里的一种。这些文件可以使用8位或16位的数据来代表每一个例子，拥有声卡的人无疑可以从任何资源里产生出这些文件。

S-Tools声音程序st-wav.exe在每8位或16位里隐藏一位信息，并且有与st-bmp.exe同样的加密选项。它也将使用一个随机数字生成器选择一个随机的位子集。这使得失真传播到整个声音文件。这个程序将显示代表声音的一幅图像并且展现给你看。数据被隐藏后，这个图显示了在红色里发生的波形变化，并且在黑色里留下了未变化的部分。这有效地揭露了隐藏部分里的1和0模型，与声音文件的最小有效位在哪些地方不同。图9.8显示了这一页的一个屏幕镜头。

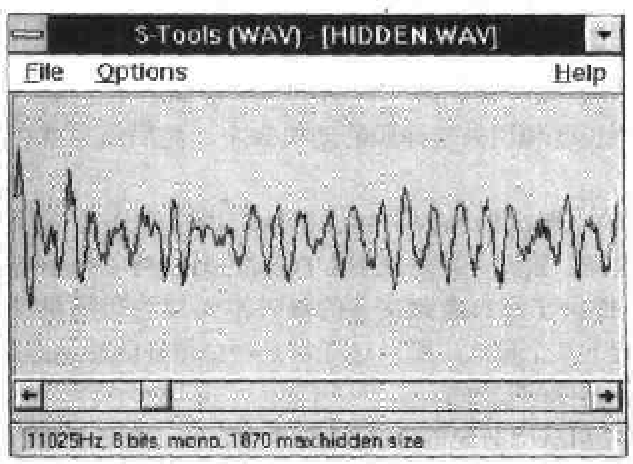


图9.8 st-wav.exe程序——在一个WAV文件里隐藏信息的S-Tools程序，这是它的主窗口，该程序显示了红色的波形变化，但这个细节不会在我们这本黑白的书中出现

9.2.5 S-Tools和磁盘真空区

S-Tools组里的第3个程序是st-fdd.exe，它将在一张软盘未分配的区域里隐藏信息。每一个盘都被分成很多段，这些段通过File Allocation Table（FAT，文件分配表）被分配到独立的文件中，如图9.9所示。没有被使用的段只是空闲着什么事情都不做。如果某些人试图用一个像文字处理软件那样的编辑程序来打开它们，或者甚至试图用一个文件管理器来检查它们，那么他们什么都找不到。这是操作系统的真空区，但是这并不意味着它不能持有任何信息，信息可以被写进这些段中并且留在里面。破坏它的惟一方法是取决于是否有某个人给磁盘写一个新的文件。操作系统会把那些段分配到另外一个文件里，因为它认为空间是自由的。

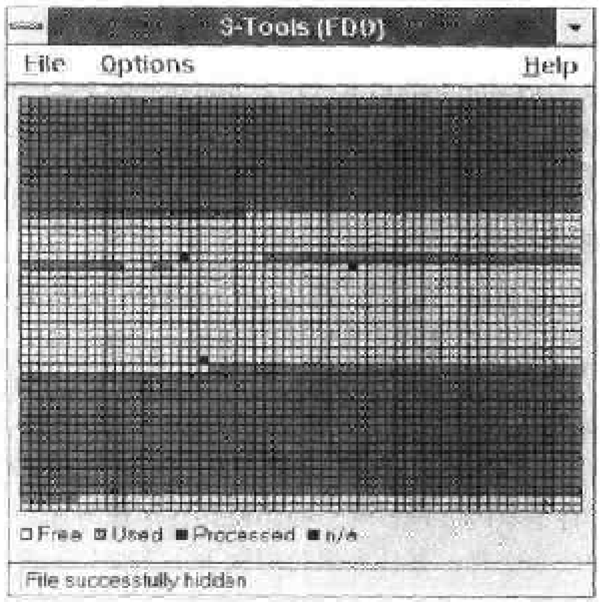


图9.9 st-fdd.exe的用户界面，显示了红色为被分配的段，灰色为未被分配的段，黄色是被转变以隐藏信息的段

S-Tools通过选择空段，并在这个自由空间里存储信息。第一个段得到了文件的标题，文件标题指明了文件长度和用来选择段的随机数字，然后信息就存储在这个随机选择的段落串里。

如果使用加密术，那么随机数字生成器就使用密钥作为一颗种子，这意味着有一个不同的随机段范围被选择。数据本身是使用S-Tools的另外两个程序里提供的5种算法被加密的。

最后，S-Tools提供了没有隐藏文件的额外空间里的写随机噪声。因为真空区可能从它存储的最后文件中留下某个模式，所以这通常是一个很好的想法。通常当一个文件被删除时，磁盘空间实际上是不会被清空的。

FAT表格里的项目从“分配的”改变到“空的”。旧的数据及其模式都仍然还在那里。这意味着某个人可以通过寻找具有随机信息的段来确认出一个软盘里包含隐藏信息的段。包

含文字文件碎片、图像碎片或普通数据碎片的段假定是无害的[①]，S-Tools将重写这些段，以便把未被分配的段转变成大量的噪声，这和使用一个新软盘是等价的。

9.2.6 随机游动

本章已经讨论了在图像或声音文件里隐藏信息，方法是抓取所有的最小有效位来负载信息。没有原因可以解释为什么所有的最小有效位都需要被利用。Hide（隐藏）和Seek（寻找）以及S-Tools都使用了随机数字生成器来选择字节，这本身实际上就是为什么要使用所有的最小有效位的原因。这个随机过程保证了分布在整个文件里的失真不太明显。它也使得攻击者很难计算出哪些位是重要的。S-Tools使用了MD-5算法保证了随机数字是加密安全的。

实际上，一个随机子集还有其他一些用途。第一，如果一个人选择一个小的、随机的子集来储存信息，那么另外一个人也同样可以这样做（仿造）。如果两个人使用不同的随机源，那么就有一小部分位有机会结束在两个子集里。误码校正代码可以帮助从这些冲突里得到恢复，这就可以允许几个人使用相同的文件把信息传送到其他几个人。

Steve Walton在他的文章*Image Authentication for a Slippery New Age*[Wal95]里提出了一种方法，这个方法使用了一个普通的、二维的随机游动，它是在一张照片的周围晃动。偶然地，路径可能会包围它本身，这需要保持对路径从前经过的地点进行跟踪。相比之下，"捉迷藏"把照片看成是一个一维像素表，并且只简单地选择像素的一个随机数字来往前跳跃。

Walton设想他的随机游动里的最小有效位可以被用来设置为图像的一个密封。就是说，可以通过在最小有效位里嵌入图像的某个数字信号来给图像"作标记"。自然地，这个数字信号只能由非最小有效位计算出来，这是因为只有那些位在整个过程中保持不变。这个密封系统可以被职业摄影师用来在一幅照片里附加上他们的标记。

有些人争论说，这个方法是一个废物，他们相信在照片的末尾添加数据信号具有更大的实践意义。这个信号类型要能够处理所有类型的照片格式，包括没有足够有效位以隐藏数据的二进制图像。同样，在编码信息时没有必要去刻意避免最小有效位，这样信号就可以更好。

> 另外一种解决方法是简单地产生随机位排列。Tuomas Aura在[Aur95]里详细地描述了这种方法。

这些建议当然是正确的，这个秘密方法的惟一优点就是秘密。摄影师给图像做标记大概就是为了保护他们的版权，他们可以确实地证明照片被盗。如果标记被添加到文件上，那么某个人可以简单地消除它或者篡改它。如果它（标记）是以随机游动在最小有效位里被隐藏，那么想盗用照片的人就不得不先找到标记。当然，为了预防怀恶意的人，可以对一张照片的最小有效位作写保护。

在整个数据中简单地选择任意的随机游动有一个简陋的方面，碰巧在同一幅图像里隐藏信息的两个人之间并不需要任何事先的交流，他们都保持几个手指头的距离以使冲突是很微小的，并且误码校正代码能够校正它们。

---

① 首先使用st-bmp.exe或st-wav.exe程序在一个照片或声音字节里隐藏信息，然后可以在未被分配的段里存储它（隐藏的信息），这样它看上去就像被任意抛弃的信息。

在一个图像里产生复合通道有一个更原则性的方法，如果所有部分都事先调整它们的用途，那么它们就可以保证它们的随机游动不会冲突。因为这样误码校正代码就用不着再被使用，这就节省了空间。但是这却增加了过程的复杂性。

为了产生n个通道，把文件分成n字节数据块，每一个数据块的一个字节都将给定一个通道。在最简单和最透彻的方法里，字节分配和通道数都是很难编码的。通道1得到字节1，通道2得到字节2，依此类推。一种更好的方法是用一组1～n之间的倒置值完全地打乱字节。这里有一个产生随机倒置排列组的好方法：

1. 以规则组{1，2，…，n}开始。
2. 为了产生一个新的随机排列，重复上个步骤j次。j更大一点更好，但是j太大会没有效果。
   a. 使用一个加密安全随机数字生成器的输出来在这个随机组里选择两项。
   b. 交换它们的位置。例如，如果交换前的组是{5，1，3，2，4}，用随机数字生成器选择第二和第四个数值，那么结果将是{5，2，3，1，4}。
3. 输出这个排列。转到第2步继续执行。

由这个序列产生程序所产生的第i个序列，可以确定在第i个数据块里哪个通道得到哪个字节。这样就保证了两个用户之间不会发生冲突。

另外一种方法甚至会更多地破坏这个过程。为什么数据块应该由相邻的字节组成？在最简单的方法里，从一个n通道系统里的通道k而来的字节i被分配到文件中的字节$in+k$，这可以使用取幂以文件的长度为模进行编码。所以如果文件是p字节长，并且恰好p是质数，那么$(in+k)^e \mod P$将编码字节使得它们不相互毗邻。

9.2.7 JPEG的使用

本章的第一部分讨论了JPEG的图像文件在最小有效位里保存数据的影响。因为有损耗的压缩算法并不在意它是否正确地重新建立了一个文件，所以它就把所有的信息干扰置入不存在状态。虽然某个人用JPEG压缩文件是一个难题，但是这并不意味着JPEG算法对那些想在图像里成功地隐藏信息的人是没有用的。

使用JPEG算法存储信息有两种可能的方法。第一种是使用它作为确认图像复杂性的工具，本节将讨论这个方法。第二种方法是使用一些标准的隐藏部分来隐藏信息，将在9.2.8节里描述。

JPEG算法是确认一个景色里的清晰度的一个很好的工具。这个清晰度可以被用来找到图像中最嘈杂的角落，数据就可以存储在这个角落里。在本章的第一部分里，隐藏信息的基本算法使用了n个最小有效位来隐藏信息。如果$n=1$，那么就将产生一个随机分布在整个图像的较小的、但却是一致的结果。如果n更大一些，那么就会有更多的信息在图像里储存起来，但是同时更大的失真也会出现。在任何情况里，失真都会贯穿整个文件而被统一地分布，即使这是不太实际的。

例如，设想有一张照片，里面有一个人坐在一片绿色草地中间的一张红白相间的格子花纹的野餐毯上。在一个绿色部分上设$n=4$是有道理的，因为它是焦点没对准的地方，没有特别装填重要的清晰度。在另一方面，因为清晰度对照片非常重要，所以在脸部区域可以使用$n=1$。当然，可以手工地仔细检查照片并且确认照片中最重要的、最脆弱的部分，但是这

不会使这个算法的意图失效。这（手工检查）不仅是耗时的，而且也必须为交流另一端的那个人构造实际上相同的部分。这是他们所知道的恢复位的惟一方法。

JPEG算法提供了一个分割照片和确认图像里最重要或最显著的部分的自动方法，它被这样设置以增加压缩。这个算法生成器调整了算法以便它能提供令视觉满意的图像，即使是在有一些清晰度已在压缩中消失。

这个应用是简单的。设f是一个等待数据在一些最小有效位里被隐藏的24位图像文件。设$JPEG^{-1}(JPEG(f))$是具有JPEG的第一个压缩f的结果，然后，解压缩它。f和$JPEG^{-1}(JPEG(f))$的不同，揭示了有多少噪声可被利用来隐藏信息。对每一个像素，可以用f和$JPEG^{-1}(JPEG(f))$相比较，并且确定有多少位是相等的。如果编码蓝色的位中只有前4位或前8位是相同的，那么就可以推断出JPEG算法并不真正在意最后4位里是什么。这个算法确定了那些可被设为任意数值的4位，图像结果看上去仍然足够好。那就意味着4位可以被利用来隐藏信息。在图像里的其他地方，所有的8位f都符合$JPEG^{-1}(JPEG(f))$。没有信息会在这些位里隐藏了。

这个算法使得确认图像最重要部分的位置是可能的。可为JPEG选择合适的精确度数值。如果需要一个最好的最终表示法，那么就应该使用最好的JPEG设置，并且这将确认出可利用来隐藏数据的一个更小的位数。一个粗糙的JPEG设置应该揭开更多的位。

还有很多其他的隐藏信息的压缩算法正在研制发展中。由Barnsley发明的重复系统（参看[BH92]）产生的分形压缩算法是一些更大众化的技术，它们中每一种算法都可以使用在相似的模型里以确认图像的贡献部分。

转变成不同类型图像的其他解决方法也可以被成功地使用。例如，有一些算法是被设计用来将24位颜色图像转变成8位颜色图像的，它们都可以很好地确认代表图像的256种颜色。可以通过把24位数值和被选择代替它的256色表格里相应的项相比较，以确认每一个像素的自由位数。这些算法中有一些被调整以更好地完成人脸部分的工作。其他的算法将用来处理自然景色。每一个它本身的方法都是可应用的，并且可很好地完成系统工作。

如果使用JPEG或一个相似的有损耗的算法来确认一个图像的高噪声区域，那么就必须改变系统的一个关键部分。当一个GIF文件被用来储存信息，那么接收者就不需要拥有一个原始图像的副本。N个最小有效位可以被简单地去掉和恢复，它们可以被逐个位地被使用。如果JPEG将指出等待放置更多数据的图像拐角和裂缝，那么发送者和接收者都必须访问相同的拐角和裂缝表格。也许完成这个的最容易的方法是确保发收双方都有原始图像的副本。如果将要和某个从来没有见到过的人交流，这就是一个限制，必须以某种方式安排使那个人得到这个原始图像。

9.2.8 在JPEG文件里隐藏信息

毫无疑问，JPEG隐藏信息的有损耗方法是一个难题，它混淆了用于隐写术的基本方法。噪声可以以任何方式改变，所以第一个脉冲应避免全部编排在一起。**Derek Upham**钻研得更深并且找到了另外一种方法，他称这种方法为*JSteg*。JPEG算法压缩数据用了两步。第一步，它把图像分成8×8的像素块并且为了描述它们，使这些像素符合余弦函数 （如图14.10所示）。就是说，它找到函数的加权平均值，这个值加起来和8×8的像素块的原始值非常接近。第二步是它储存这些余弦函数的加权平均值作为这个像素块的描述，通过把更多或更少的加

权值设为0，压缩量可以被相应地增加或减小。Upham认为，可以通过调整加权值的最小有效位来储存信息。

他的解决方法用C语言编码并且作为一个diff文件发布，这个diff文件可以被加到在网络中发布的标准JPEG版本4里。他的代码为UNIX加入了一个额外的直线命令功能，它允许在压缩一个图像的同时隐藏一个文件。因为它简单地建立了标准JPEG分布，所以它是一个很好的方法。同样，因为JPEG图像格式在网络上是很普遍的，所以它是很重要的。它比照片的GIF格式更有效，尽管GIF通常在图解上更好。

为什么这种方法可以实际地产生比简单调整数据的最小有效位更好的结果，这也有很多生理学的原因。像S-Tools和Stego那样的程序跳过了很多循环来处理8位色图像，它们是以在颜色表里的一群颜色结束的，这些颜色相互都非常相似，这些工作可以很好地完成，但是却很容易被扫描颜色表的某个人发现。

调整模型化8×8数据块的离散余弦变换的频率会有一个不同的结果。虽然这些调整会危害到最终图像的质量，但是要区分它们的结果是很难的。毕竟，离散变换值已经是一个近似值了，而且要注意到，想发生在一个近似值里的变化是更加困难的。在本质上，位是通过控制JPEG程序是否舍入或舍去而被隐藏的。舍入是一个1，舍去是一个0，这些数字可以通过查找最小有效位的频率被恢复。

Upham选择了一个令人感兴趣的方法来隐藏信息。在数据块的开头必须有一个标题来说出那里有多少位。通常，这是一个单独的数字。所以前32位应被归纳为一个可以暗示它们的位置的数字，比方说，8523个字节存储在紧跟其后的最小有效位里。Upham注解到，这个数字出现时通常前面有很多个0。既然这些位从某种意义上来说是标准随机分布的，那么一个0数据块看上去就很可疑。一行里的16个0出现的机率应该只有1/216。

他的解决方法是把标题分成两个区域。第一个区域由一个指明第二个区域里位数的5位数字构成。第二个区域包含了整个文件里的位数。这样就清除了数字开头的任何大位块，给极大的文件留下了一定的灵活性。他也建议第二个区域里的位数应该在中间时刻添补一个额外的数字0，这就阻止了文件的第6位总是一个1。这是一个很微妙的注意细节。

9.2.9 智胜

由Niels Provos编写的“智胜”软件像所有其他的工具那样可以调整最小有效位，但是这个软件在调整时非常仔细，这样就可以避免引入一些能提醒攻击者寻找信息存在的统计信号。

对统计细节的注意力也会有它的代价。这个程序确认潜在位，并且为了拥有潜在的调整，从而它取消使用半数的位。这就把通道容量分成了两半，但是却剧烈地增加了不安全性。

在大多数抽象的意义上，猜透算法都是很直接的。每次改变一位来隐藏信息，就要寻找到一个等价的位，并且大大地改变它以维持一个平衡的统计轮廓。如果把一个0改变为一个1，那么同时也要改变一个1为一个0。若给出了确认隐藏数据的好位置的程序，就可以修改该软件以适应任何数据格式（参看[Pro01，Pro]）。

这些平衡的变化可以在任何加密方法中使用。在实践中，Provos通过改变JPEG压缩系数来执行这个算法。简单地改变最小有效位会引入更高位的JPEG压缩系数的变化，并且这

些改变可以相对比较简单地被察觉（参看[JJ98a]）。

9.2.10　F4和F5

JSteg是第一代设计用来在JPEG图像格式里隐藏信息的程序，但攻击者很快就发现了它留下重大的统计量特征。JPEG压缩的参数通常向前下落了一个钟形曲线，信息隐藏过程使它们失真。在大多数图片里，最标准的参数是0跟在1和－1后面，再跟在2和－2后面，并依此类推。这些参数下落出现的原因是参数取得太大了。

当隐藏信息时，参数破坏了这个平滑曲线。例如参数为2和3，在调整最小有效位的过程中被互换，以同等比例将它们留下[JJ98a，JJ98b，WP99，Wes01]。相同的过程发生在参数对4和5或者－1和－2中。如果发现在有许多参数对的JPEG文件中出现相等的情况，就标志着它可能使用了JSteg算法。

在隐藏信息的时候，一个方案可以改变0和1的这种可能性，第6章的算法能够通过模仿一些分布来改变这种统计。如果说，我们对一个JPEG图片的分析显示，相对参数2来说，参数3的出现几率是60%。一个3相应一个隐藏1的位，而2相应隐藏0的位。我们安排了隐藏的1的数量相对0的数量来说是60%，这样参数就失去了平衡。

这儿有一个基本解决方案：创建一个n位的字的集合，像6.1节里的算法一样使用它们。在文字中使用0和1的数量来决定权重的大小。如果0分配给权重.625，而1分配给权重.375，这样数值就显示了近似的分布为1比0.6。

举个例子，令$n=8$，在字母表中就有256个字符。赋权重00000000为8×.625，00000001为7×.625＋.375，等等。这些被用来建立一个霍夫曼树形结构来改变输入数据的统计量。如果进行了这些预处理，JSteg算法的统计量就很接近真正的图像了。这并不是完美的，但它也不需要这么完美，因为参数的数字在每个图像上都是变化的。

Andreas Westfeld建议了统计量差异的另一个解决方案，他的算法称为F4，能更小心地加密数据，避免了统计失真[Wes01]。

下面是用系数C隐藏数据的技巧：

- 如果$C=0$，略过它，且不用这个参数藏任何信息。这样减少了算法的效率，但是没有别的选择，JSteg也是这么做的。
- 如果C>0且为奇数，不改变C，并用它来编码一个1，并递减它编码一个0。
- 如果C>0且为偶数，不改变C，并用它来编码一个0，并递减它编码一个1。
- 如果C<0且为奇数，不改变C，并用它来隐藏一个0，并递增它隐藏一个1。
- 如果C<0且为偶数，不改变C，并用它来隐藏一个1，并递增它隐藏一个0。

这只出现了一个问题，当递减或递增的时候产生新的参数0。在此情况下，这个位被重复，因为信息实际上丢失了。编码过程会忽视为0的系数。

表9.3展示了略微不同的编码信息的技术。

表9.3 用F4调整参数方案编码位

系数	对0编码后	对1编码后
0	略过	略过
1	0（略过）	1
2	2	1
3	2	3
4	4	3
-1	-1	0（略过）
-2	-1	-2
-3	-3	-2
-4	-3	-4

表9.4表现了位01010被隐藏。这5位需要7个参数，因为有些被略过了。

表9.4 用表9.3编码01010的一个例子

隐藏位	系数前	系数后
0	4	4
1	2	1
0（略过）	0	0
0（再次）	-2	-1
1	1	1
0（略过）	1	0
0（再次）	2	2

编码文件是简单的。例如，一个参数1，如果不是由用参数2隐藏一个1，就是用参数1隐藏一个1。在任何一种情况下，被隐藏的位是1，类似的情况贯穿整个加密过程。

这种解决方案避免了Jsteg算法大体上的统计问题，但仍然改变了文件。一些位通过缩短参数的绝对值来隐藏，这就意味着将会比以前有更多的数字串在0的周围，而分布现在变得更加紧密了。这并不显眼，因为在图像中分布自然地变化，但它仍然值得讨论。

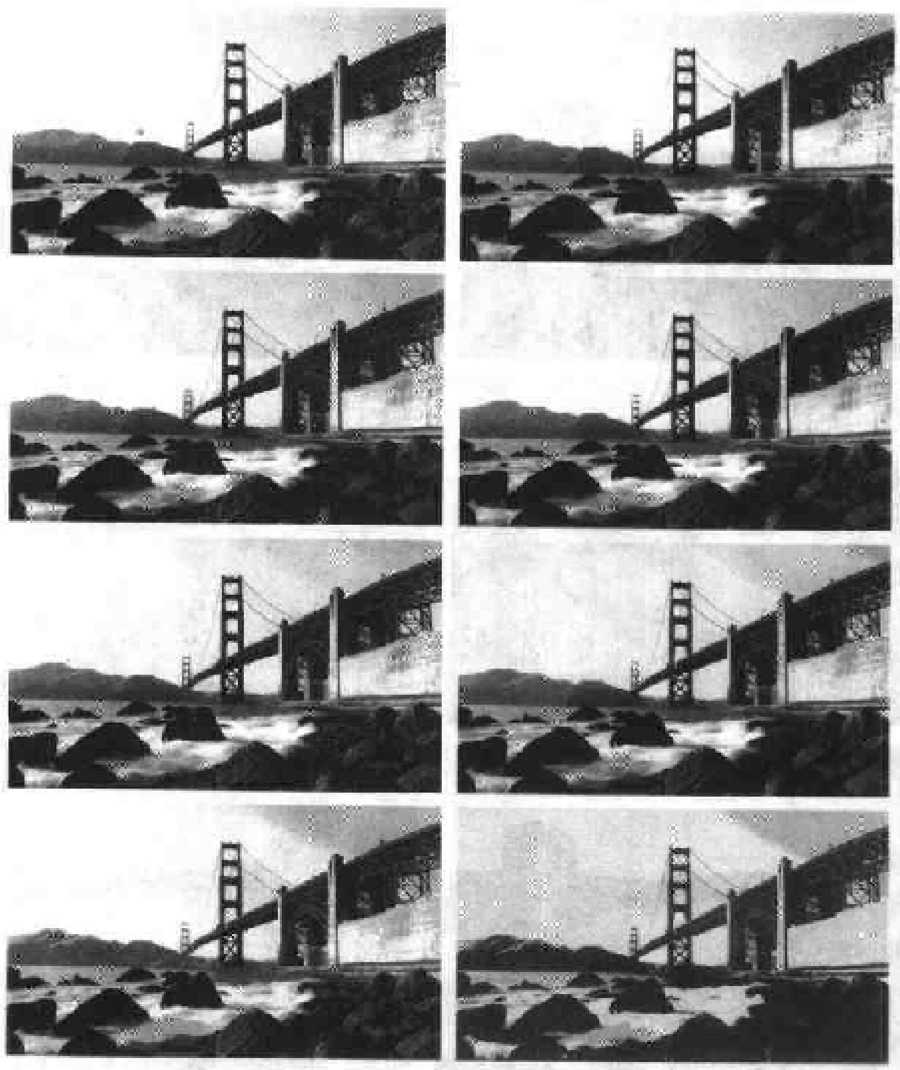
在下一个方案的算法F5中，Wesfeld更进一步地增强了使文件破坏降低到最小化的过程。他使用一个自称为“矩阵编码”的过程来分布信息而不需要更多的位，这样就减小了密度和失真的数量。

想像一下你需要存储$n=4$位的数据，一种解决方案就是在图像中挑选出$2^n=16$个不同的位置，只改变位置中的一个。一个在位置3＝0011的改变，意味着需要存储信息0011。F5算法选择最好的n值来调节被储藏的数据，用一组密码保密伪随机位流来选择存储位置。

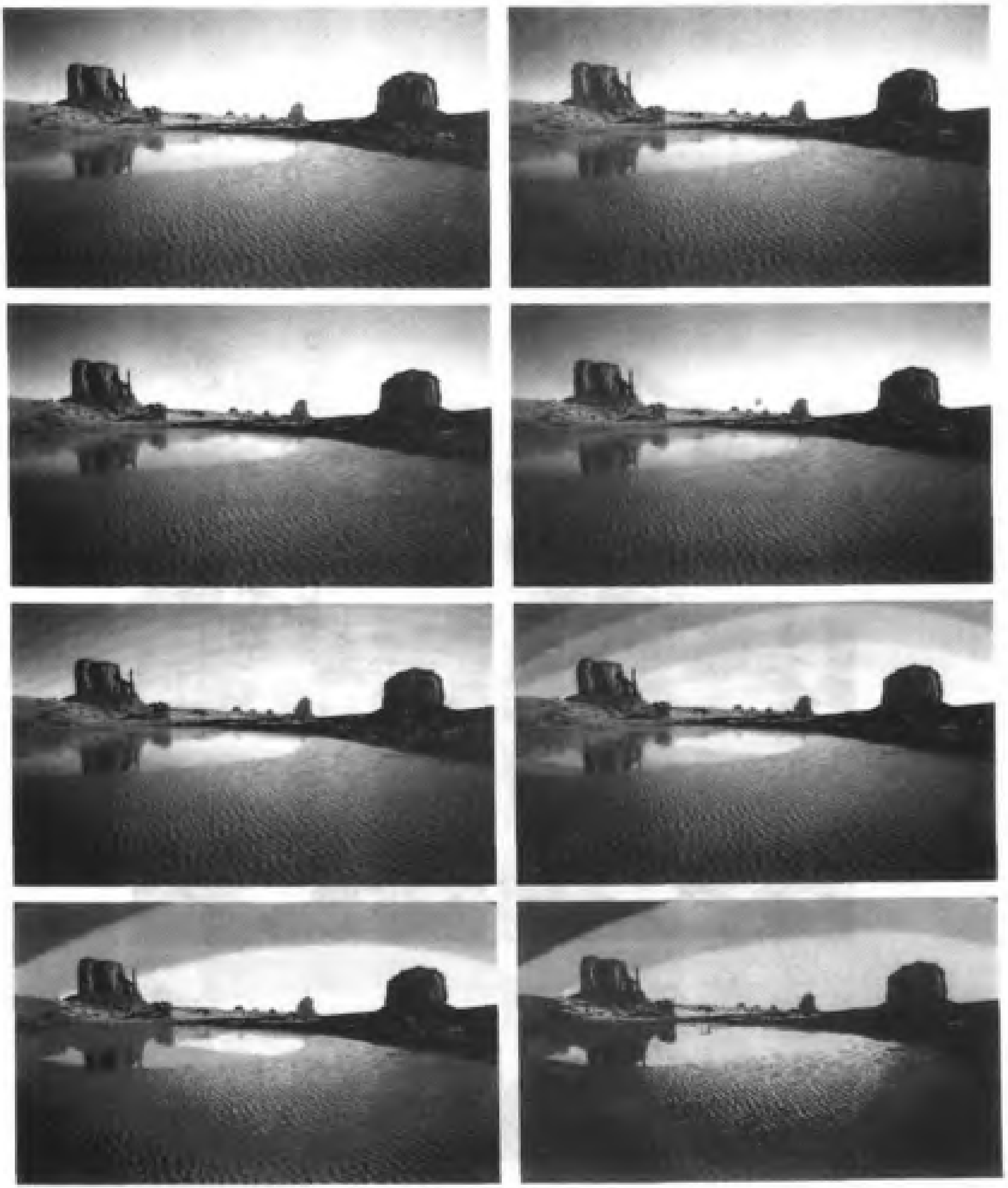
9.3 小结

在数字图像的噪声中置放信息是隐写术最广泛使用的一种方法。如果关心如何使用这些工具，利用不同的方法将保证数据难以发现。最大的问题就是确定正确地掌握了24位图像和8位图像之间的差别。

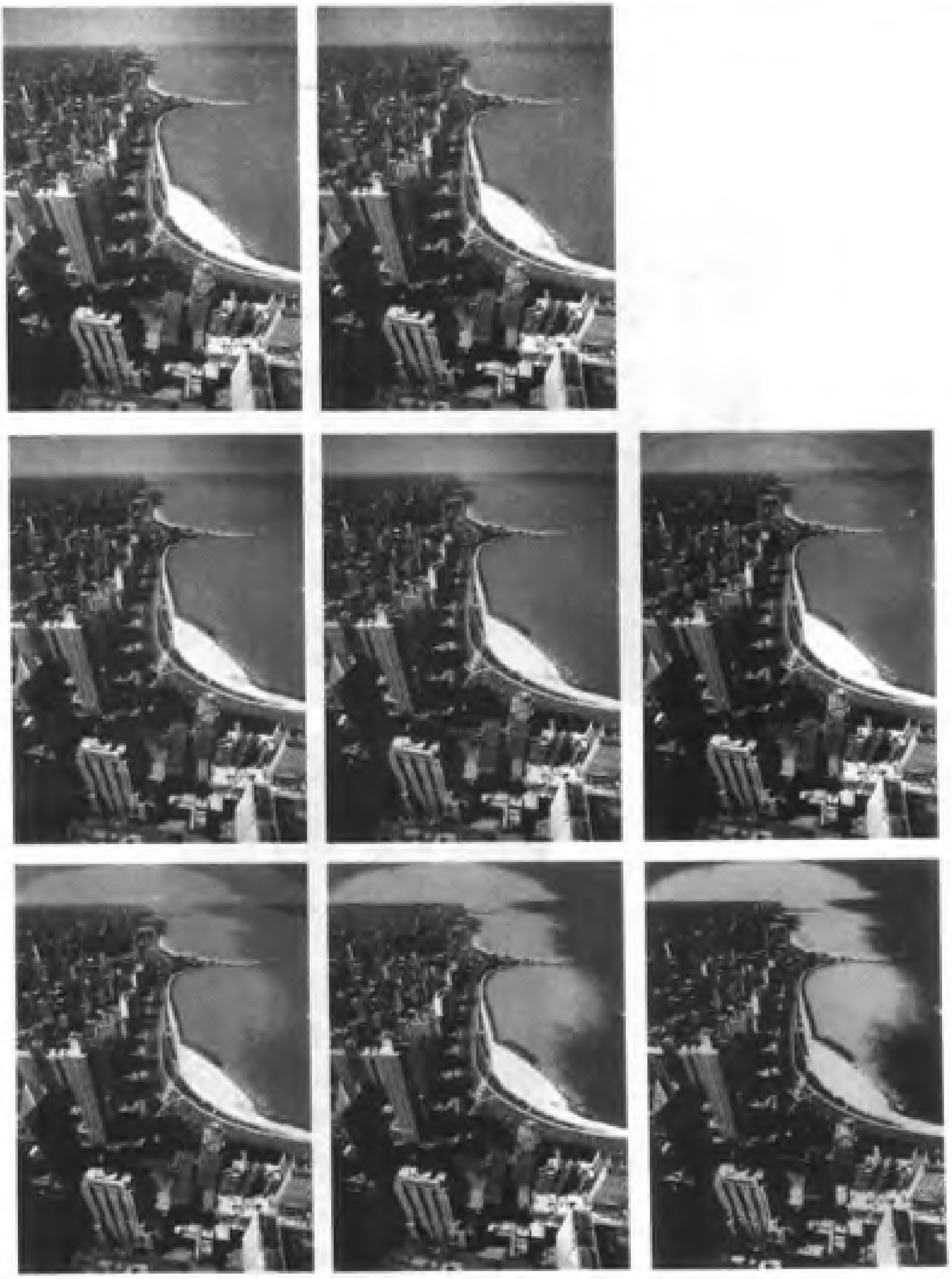
- **伪装** 世界充满了噪声，为什么许多随机性当中的一些不能被用来隐藏信息，这是不用问原因的。伪装很难被一般人所留意。
- **安全性** 如果有人知道了信息，这些系统是不安全的。但许多系统可以从源图像产生难以区分的新图像。如果数据在隐藏时没有被压缩和加密，就很难知道是否有数据隐藏在里面。第17章讨论了侦测隐藏信息存在的几种方法，如果在文件细节增添一个特别的报头就可以破坏掉这些方法。
- **怎样使用** 在网络上对这些软件的应用有许多不同的方案（见附录D） 有些在整个网络都有传播。这些程序简单易用。



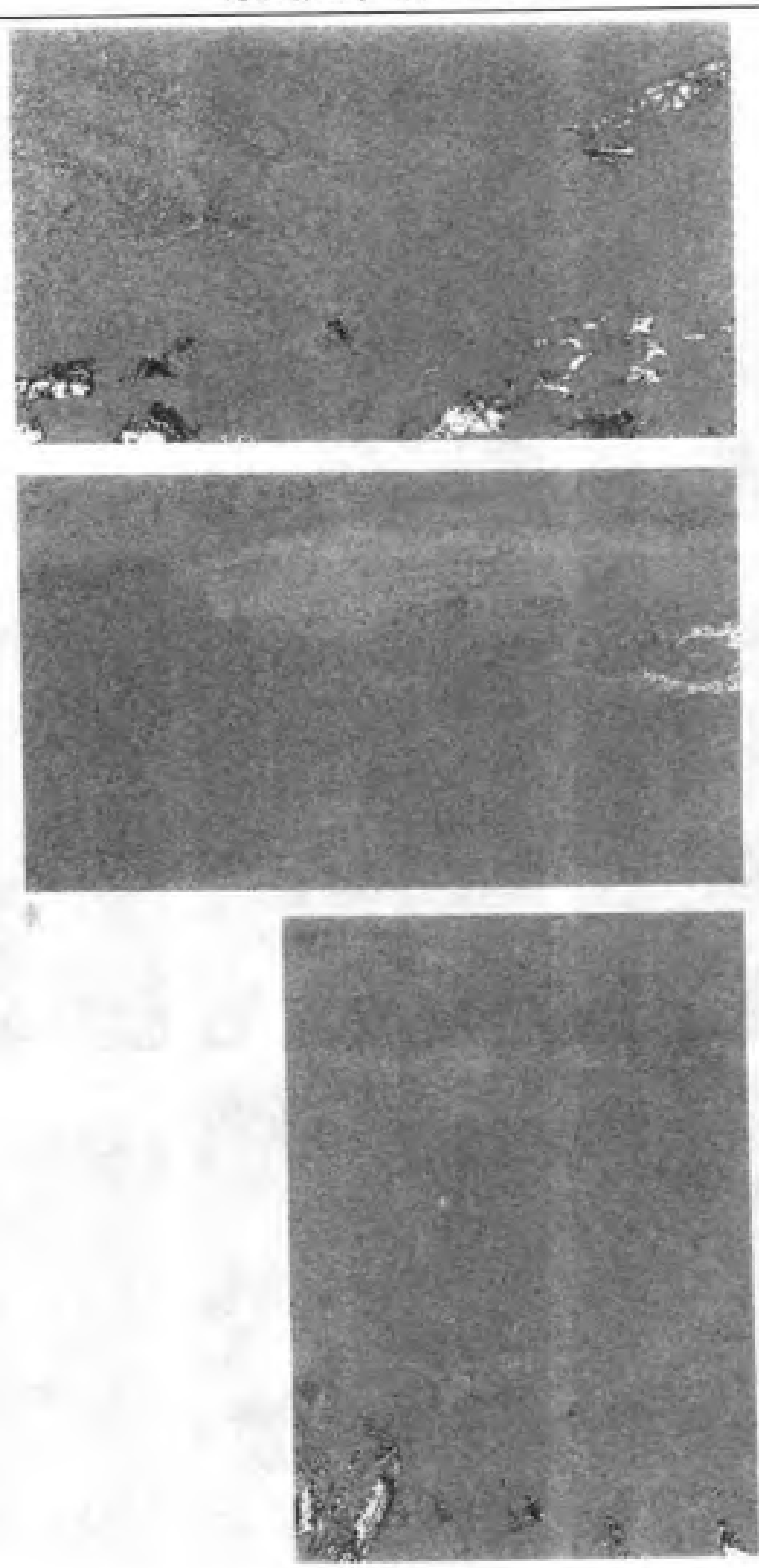
图版 I 一张图片和用一个隐藏文件替换连续位平面的结果。该图片是2312×1526像素或者说总共有3 528 112个像素。每个像素有三种成分（红色、绿色和蓝色），这样这个位平面就有3×3 528 112=10 584 336位，也就是1 323 042字节或者1.3兆字节。左上角的是原始图片，右上角是最小有效位平面被1.3兆字节数据替换以后的图片。左边第二行则表示了最小有效位平面被2.6兆字节数据替换后的结果，随后的图片依此类推。首先的影响表现在天空上。大致结构的分辨率在隐藏这些结果方面表现得非常的好（所有的图片都转换为CMYK四分色印刷）



图版 II　另一张与图版 I 有相同维数的图片



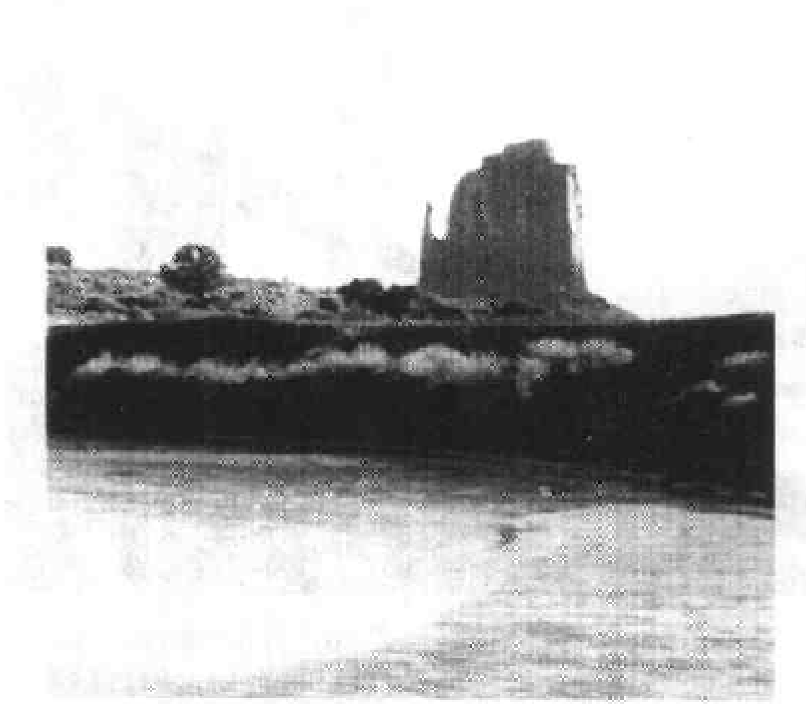
图版 III　一张纵向的1526×2312像素的图片以及替换连续位平面的结果。每个位平面储藏有1.3兆字节



图版Ⅳ　图版Ⅰ～图版Ⅲ中自然最小有效位。除了某些位置颜色比较饱和以外，很多地方看起来都是很随意的。在某些情况下，只有一种颜色饱和而其他颜色都不饱和



图版Ⅴ 图版Ⅰ所示形式的图像用异或来比较它的最小有效位和最大有效位。大多数图像是随机出现的，但是有意义的协定可以在大多数饱和或不饱和的点上被发现。用一个破坏这种关系的信息来代替最小有效位



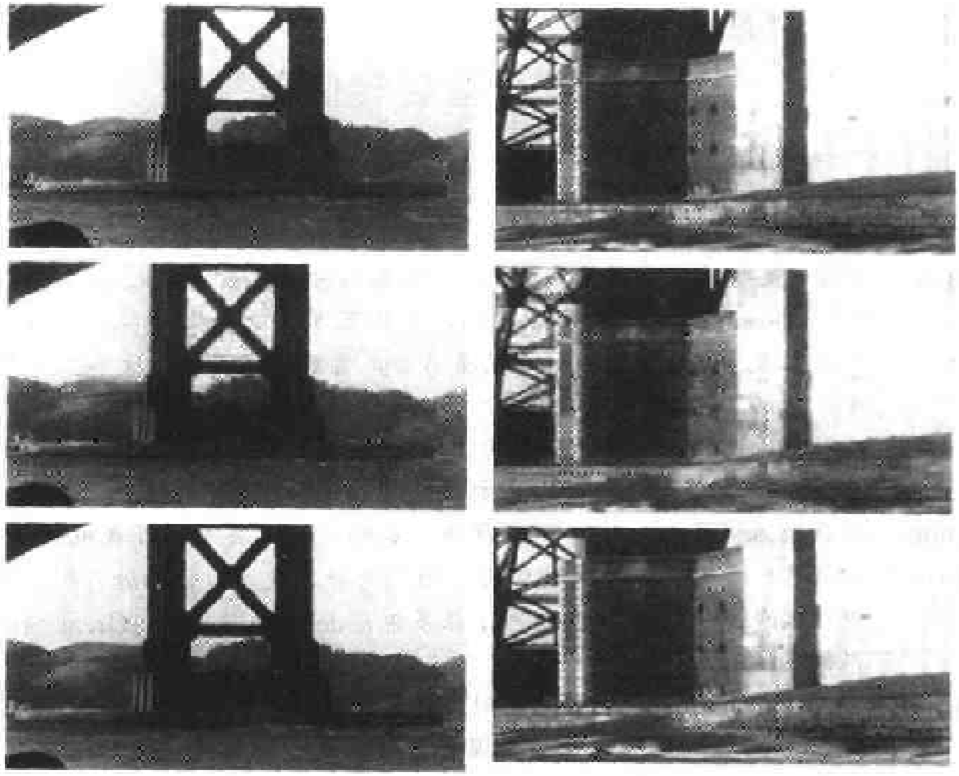
图版Ⅵ 打包分组过多的数据会留下人为的痕迹。这张2048×1352像素的图片显示了在其四处最小有效位被ASCII文件置换的分辨率。ASCII码的第八位总是为大写字母0，第七位总是为小写字母1，这就造成了每隔八行就会产生垂直的条纹



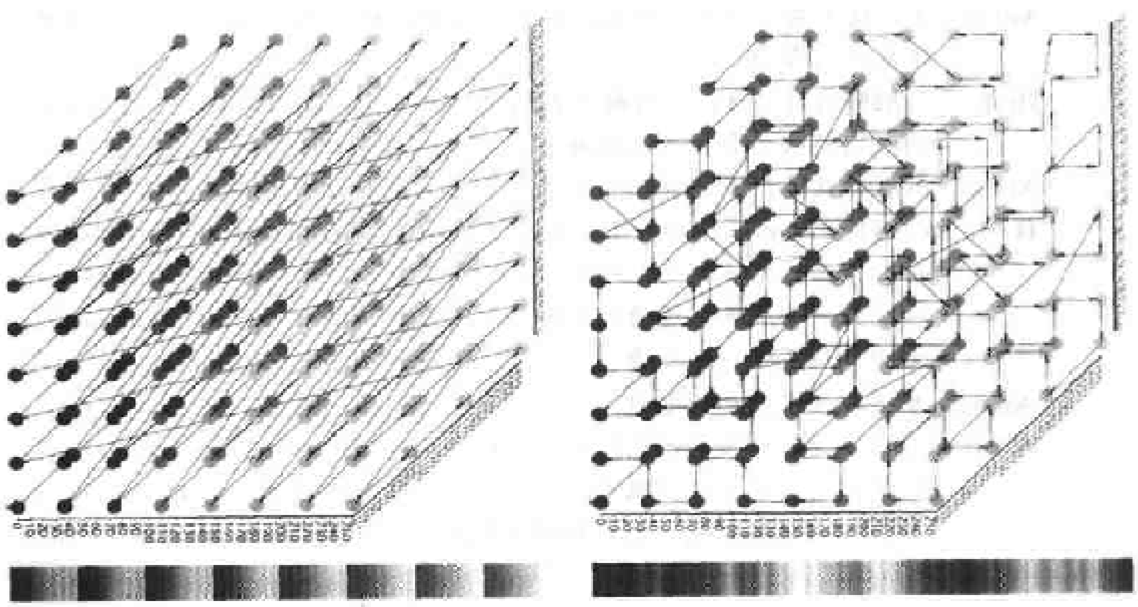
图版Ⅶ 左边的窗口表现了一张图片和图像中被EzStego隐藏大约17 000字节前的最小有效位，图片的容量为20 000字节。右边的窗口则显示了这张图片和编后记的最小有效位（这是图17.2的色彩形式）



图版Ⅷ 一个大的被F5算法填充数据后的低质量（10）JPEG文件模型。这个算法沿着天空中蓝色的调色板留下了粗糙的直线。当低质量和高压缩的图片携带过多的隐藏信息时，系数的细微变化也变得非常明显。在低压缩和携带少量数据的图片上，这些人为痕迹就消失了



图版Ⅸ 最上一排的图片显示了没有加入信息的低质量（10）JPEG文件的分辨率。中间一行图片显示了用F5算法混合最大信息数据后的结果，系数的细微变化在分辨率上仍然是可见的。最底行的图片表现了在高质量（100）下用F5算法填充数据后的结果。在低质量下填充太多的信息会留下人为的痕迹



图版X 上面的图是一个普通的调色板，而下面的图是被整理过的调色板形式（这些图形由Andreas Westfeld和Andreas Pfitzmann在他们的论文[WP99]建立的）。他们在附言中只用了61根短线（这是图9.7的一种色彩形式）

第10章 匿名转信器[①]

匿名先生和你的谈话

Host: 在这个星期的展示会上，我们有一位匿名者，他在像Madonna、Michelangelo和早期的Prince这样的艺术家之列，这些艺术家是如此出名，以至于他们可以靠自己生活。这位男的或是女的匿名者，曾经读过世界上大多数具有煽动性的著作，能把他（她）请到我们今天的展示会上我们感到非常荣幸，即使他（她）只同意通过朦胧的电视链接出现。匿名先生（或者我应该说匿名女士？），很高兴您能来到我们今天的展示会！

Anon: 叫我匿名博士吧，这样可以解决性别的问题。去年六月我被授予名誉博士。

Host: 祝贺你！那是一个很高的荣誉。他们选择你是因为你的著作吗？在我这张简明概述的纸条上注明你已经写了很多像*Federalist Papers*，*Great Stuff*这样的热情浪漫的小说。

Anon: 实际上，Federalist Papers直到资料收集完后才能成为一本书，这真的不是一本浪漫式的书，虽然它在关于国会是什么这方面有更现实的想法。

Host: 两性的国会。这是我暂时没听说过的委婉的说法，你是来自于那所古老的学校，是吗？那里是你获得学位的地方，是吗？

Anon: 是……

Host: 这可以成为你为什么犹豫是否与观众见面的解释，是吗？这太虚饰了。

Anon: 不，这不是我的生活方式，我宁愿保密我的身份，是因为我写的东西有一些有很危险的影响。

Host: 多聪明的计划呀！你引起了我们极大的兴趣，我相信每一个其他读者都会加入到我们的展示会来，我们非常高兴能和你面对面地交谈。

Anon: 我有点犹豫，但是我的出版商坚持要这样，这是合同的内容。

Host: 你觉得在现代社会做个名人难吗？通过和Cher做习题书籍会增加你的曝光程度，对此你感到有压力吗？她也给自己取了个名，你们一定会相处得很好，你可能会谈到那个汽车公司的记录员如何使你陷入艰难的时刻，因为在前面开始的表格中你留下了一个空白。

Anon: 哦！我还没有那样想过。

Host: 对现在媒体传播的结构体系和风格，你觉得怎么样？他们总喜欢拍摄那些活着的名人的私人生活的照片。你觉得做富裕和有名的人的生活方式怎么样？他们将把你的生活方式、地点和过程展现给每个人看，这真是出卖人格的好方法。

---

①译者："匿名转信器"是我们对"Anonymous Remailers"的中文译法，也有译做"匿名重邮器"的。Anonymous Remailers是一种隐密的电子邮件服务，它允许你寄出电子邮件给新闻群组或是某个人，但接收者将不会知道你的姓名或你的电子邮件地址。互联网上有许多不收任何费用的提供匿名转信器的网站。

Anon： 实际上，我保持匿名的部分原因是为了没有人在半夜还出现在我的家门口。

Host： 哦！是呀，他们老是这样，我也遇到过这种问题。

Anon： 这和烧房子杀人没什么区别。

Host： 哦！是呀，我还见过你和Martha Stewart就如何举办一个成功的化装舞会出书，你们可以做一些真正可爱的面具并在斋月的最后一天在新奥尔良投放到市场上，你有过这样的想法吗？

Anon： 不，可能要等到我提升完我最近的一本书才行，这本书就放在你面前的桌面上，它暴露了一个很隐蔽地榨取人民的阴谋，金钱从税收账目中分离出来注入了私自合伙人的网络，这个网络是富裕的人填满自己保险箱的地方。

Host： 你认为现在我们这种谈话节目怎么样？我想我不应该申请竞赛，但你的主机像幽灵一般，你可以带着一块黑色的大头巾和斗篷在观众中闲庭信步，就像在Mozart的电影里一样。也许他们可以用电子处理方法加深你的声音，使你在谴责他们诡计的时候每个人都害怕你，就像在*Wizard of OZ*里一样。在太阳底下穿长袍那将非常炎热，但是我可以看见你已经吸引了一大部分白天的观众，你将是不同的。

Anon： 我的书真正指示了一条通往革命的道路，里面命名了很多名字，它展现了钱是怎样流通的，它指出了哪些政客是这个网络的部分，它展现了哪个媒体集团生产出糖衣炮弹使得每个人都像梦游一样。

Host： 哦，你没有发现Anonymous这个名字太长了点吗？你直接写"Anon"怎样？如果我叫你"Anon"你会觉得不舒服吗？或者我应该叫你"Dr. Anon"？

Anon： 两个都可以，我不介意。

Host： 我想我不应该这样叫，怎么能设想像这样厚的一本书不把你的全名写上去。你赛过马吗？（赛马填表是必须写全名。）我想问你你，是否就是Carly Simon歌曲中的那个人？"你是如此的徒劳，你可能想这歌曲说的是你。"她没有告诉任何人这首歌讲的是谁，我想可能是你。包括所有的秘密的事情。

10.1　匿名转信器

为什么人们在写信或在BBS中发布公告的同时不想附上自己的名字，其中有很多方面的原因。有些人通过匿名自杀预防中心来寻找忠告或建议，另外一些人则想在不伤害自己名誉的前提下寻求工作。还有些时候，我们尽全力促使自己写一些冗长的、半疯狂的文章，这些文章都是围绕着那些有权利的人不想听到的真相来写的。这只是匿名地发送信息的一小部分原因。即使是一个帮助政府计划对密码技术采取严厉取缔措施，以及对我们日常生活强加命令的政府高级官员，他或她也会时常地使用投币电话，这仅仅是为了匿名而已。

我们正在做的大多数事情，或者我们过去曾经做过的事，很多都是采取匿名的方式。跟踪什么人在什么时候做了什么事情是浪费时间。人们只记录有意义的数据，剩下的很快就被忘记，方式是通过提供一片宽恕来覆盖过去。

在因特网上，匿名转信器是人们匿名地相互交流的简单的解决方法，就是拥有用新地址接收即将到来的邮件和一些信的实体文本的这种功能的邮件程序。这种程序可以剥去包含

发送者真正身份的标题并且把这些内容重发给新的地址，这样接收者就只知道这是来自匿名转信器的邮件，但他们不知道是谁把它发送到那里。

在某些情况下，匿名转信器为发出的邮件产生一个新的伪匿名地址，这有可能是像“an41234”这样的随机的排列。然后，它将保存一个内部秘密文档，它会使这个随机名字和邮件的真正名字相匹配。如果接收者想回复发信者，他们可以发送邮件回“an41234”地址并通过匿名转信器转交到重新装配和发送信件的人，这样就允许人们通过邮件来进行交谈而不用知道对方的真实身份。

还有很多像这样的合法的服务要求。刊登个人广告的报纸大多数也只提供匿名邮件信箱和语音信箱，以便他们可以掩蔽自己的回复。如果匿名邮件信箱能够给他们提供保护措施的话，人们将很愿意在报纸上为找新朋友和爱人而刊登广告。有些人在觉得足够信任和别人见面之前可能更改过几个信箱地址，或者他们会用投币电话打匿名电话。数据世界已经有足够的故事说明匿名见面造成的悲剧，但这是现代生活的必需的特征①。

当然，还有很多有关使用匿名转信器的有争议的方法。有些人用欧洲标准的匿名转信器从科学教会发送版权文件到因特网上，这引起了教会的愤怒，他们能够使当地警察袭击这个地点并强迫所有者揭露出发送者的名字。显然，转信器可以用来发送损害名誉的的信息和伪造的文件，包括法庭文件和其他秘密信息。跟踪犯人取决于转信器所有者能否很好地保守这个秘密。

因特网上有很多各种各样的匿名转信器，所收集到的匿名转信器不断地增多和减少。一个当前好用的转信器清单可在**http://anon.efga.org/Remailer**和**http://www.chez.com/frogadmin**找到，包括软件指示器和关于启动你自己的匿名转信器的方法。

10.1.1 提高

有许多不同的方法能使匿名转信器提高并使其具有不同的特性，以下列出其中最重要的一些方法：

- **加密** 转信器有它自己的一对公共密钥，并且以密码形式接受请求，在发送前先得解密，这是防止侵入转信器输入线和输出线的重要措施。
- **延时** 为了使任何想偷窥转信器输入和输出通信内容的人产生混淆，转信器将会延时地发送邮件。这个延迟可由输入信息指定或者任意分配。
- **填充** 偷窥转信器输入和输出通信内容的人能通过比较大小来追踪加密信息，即使输入和输出信息由不同的密钥加密，它们的大小仍然一样。用随机数据填充信息可以解决这个问题。
- **重新排序** 转信器以一种顺序得到信息，但在处理信息时并不是按第一个输入就第一个输出的顺序。这添加了一个附加的隐秘措施。
- **转信器链接** 如果有一个匿名转信器崩溃并泄露了你的身份，为了额外的保密而把几个转信器链接在一起是有可能的。这种链路和它最强的链接一样强大，而并不像比喻中的物理层那样，只有表格中的机器才不得不保持秘密并停止跟踪。

①真够奇怪，在过去人们依赖于更多知道别人的情况来防范这种类型的背叛。比如小镇上的人相互认识，也知道每个人的名声。但是，这类的知识在大城市里并没有实际用途，所以完全匿名是最好的防范方法。

- **匿名发信器** 这个机器将以匿名地发送内容到一个新闻组来代替发送它们到电子邮件VIA（Versatile Interface Adapter，通用接口适配器）。

每一个特征都可以在不同的转信器中找到，可以在网络上查阅转信器的列表以决定哪些特性对自己有用。

10.1.2 使用转信器

因特网上有几种不同类型的匿名转信器，它们之间有细微的不同，每种类型由不同的人编写，他们用自己的方式来处理一些细节。整个概念不是太复杂，所以没有人在阅读转信器说明书后不知道怎样发送信息。

在曾经流行的转信器中，有一种是由芬兰赫尔辛基的**Johan Helsingius**创建的，他建立这个东西的时候由于法律问题耗尽了他所有的耐心，转信器地址是**anon@anon.penet.fi**。编写电子邮件和通过转信器发送它是很简单的，写一封信，把地址写成了**anon@anon.penet.fi**，在信的顶端添加两行字段，即**X-Anon-Password:**和**X-Anon-To:**。第一行是用来控制匿名身份的密码，第二行给出了需要发送的信息的地址。下面是一个小例子：

```
Mime-version: 1.0
Content-type:text/plain; charset="us-ascii"
Data: Tue,5 Dec 1995 09:07:07 -0500
To: Anon@Anon.penet.fi
From: pcw@flyzone.com(Peter Wayner)
Subject: Echo Homo

X-Anon-Password: swordfish
X-Anon-To: pcw@access.digex.net

Le nom de plume de la rose est <<Pink Flamingo.>>
```

这样，当上述信息抵达芬兰的时候，转信器会脱去标题并且把匿名ID分配到电子邮箱地址**pcw@flyzone.come**。真名和匿名被放在同一个表格中并且用一个密码加以限制，密码并不一定需要，但是这增加安全。任何有一点技术专长的人都可以伪造邮件以使它看上去像是从其他地方或其他人那里到达的。密码可以阻止任何人捕获你的秘密身份，如果人们不知道你的密码，那么他们就不能断定你的身份。

密码对消隐身份也是很有必要的。如果想从系统中删除名字和匿名身份，那么就必须要知道密码。因为转信器不想有人进入发送匿名的东西后就逃离，所以转信器就会在注销前先放置一个等待周期。基本原理就是，如果要发送一些东西，那么你就应该感觉到压力，其实这个基本原理是很受欢迎的。

如果想发送一个匿名邮件到一个新闻组，那么可以简单地把新闻组的名字填到**X-Anon-To:**，就像这样：

```
Mime-Verion: 1.0
Content-Type: text/plain;charset=us-ascii"
Date:tue,5 Doc 1995 09:07:07 -0500
To: anon@anon.penet.fi
From: pcw@flyzone.com (Peter Wayner)
Subject: Stupidity
```

```
X-Anon-Password: swcrdfish
X-Anon-To: alt.flame

In < 412A923124@whitehouse.gov> , Harry Hstar writes:
>Why you're so dumb, I can't believe that someone
>taught you how to type

You're so stupid, that you probably don't understand why this is such a
great insult to you.
```

它可以在匿名身份下发送出去，如果某人想回复，他就要通过同一个转信器回复，这可以在某些程度上保护了身份。

10.1.3 使用Private Idaho

一个在Windows平台下比较好的电子邮件程序是Private Idaho，如图10.1所示，它是由Joel McNamara编写的。其原始程序作为免费软件仍然是可利用的，但是现在要使用它的高级功能的话就必须交30美元的注册费（网站地址是http://www.itech.net.au/pi）。

Private Idaho就是一个编发电子邮件信息以及用必要的步骤加密信息的处理程序。最终的产品既可以直接发送到SMTP服务器（运行简单电子邮件传输协议的服务器）也可以被发送到其他邮件发送器，如Eudora（一种电子邮件软件，由Qualcomm公司出品）。而原始产品只能处理一些简单的工作，包括选择路径、处理密钥及加密信息。最近的版本基本上是有完全功能的。

可以从http://www.eskimo.com/joelm/pl.html得到原始版本的拷贝，从http://www.itech.net.au/pi得到最新的版本。

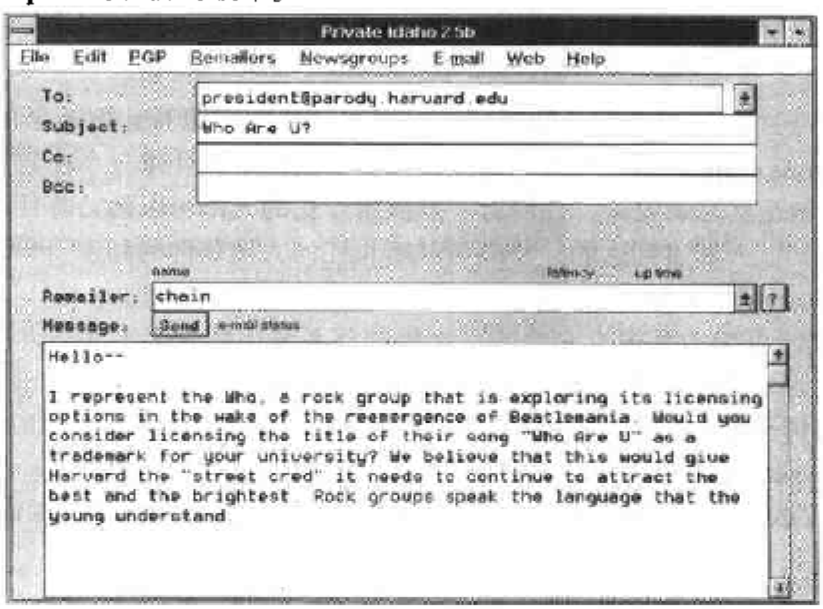


图10.1 一幅Windows系统下的Private Idaho的界面截图，可以使用它发送加密邮件或匿名邮件

10.1.4 Web转信器

许多Web站点都提供了可重新发送信息的转信器。添加一个匿名层对Web设计者来说是非常容易的事，而且很多也是这么做的。一些最流行的网站现在都是收费服务。匿名者网站（http://www.anonymizer.com）提供了发送匿名邮件和匿名浏览网页的工具，他们故意保留可以揭开神秘面纱的一些日志文件 。2001年9月11日，美国世贸中心被攻击，该公司通过帮助匿名举报者告发恐怖分子而名声大震。站点上讨论恐怖分子残害数千人实在是太冷血无情了，他们会毫不犹豫地追踪并杀害告发他们的人。

很多免费使用的邮件服务如Hotmail、Popmail、Excite和Netscape的人，都很容易丢失邮箱。这些服务在基础环境好的时候工作还可以，但他们常保留大量暴露用户的身份的日志文件。另一些，如微软的Hotmail，推出了例如Passport用来分配固定的身份给用户。

10.2 转信器的实质

设计一个转信器的内部结构是相当简单的。大多数UNIX邮件系统将取输入邮件并且使它沿着一个必需的解码程序传送，再打包分组是对标题的再排序和信息的再加密。这个过程可以通过一些简单的脚本语言和C程序块来实现。移动这些到任何平台都是很容易的，例如，有一个以苹果机的网络邮件服务器（AIMS）和著名的麦金托什机（苹果机Mac）的STMP兼容邮件程序，现在它就可以自由地发送。

转信器设计地越好、越机灵，它所面对的挑战就越大。这里有几种可能会被人们用来跟踪Web转信器信息的标准攻击方法：

- **收发追踪** 攻击者观察输入和输出转信器的信息并且以顺序和尺寸来匹配它们。对此的防范就是把*n*条信息保持在内部队列中，然后以任意的顺序分配它们。在每一个循环中，信息都可以全部保持同样尺寸或者被随机地填充。
- **转信器溢出** 设想一下转信器接受一封信而攻击者想知道它被送到哪里，这个转信器保持*n*封信在队列里，并且以随机顺序分配它们。攻击就是在考虑范围内的信息到达之前，攻击者发送*n*条自己的信息到转信器，由于攻击者知道自己的*n*条信息的目的地，所以攻击者就能够从信息流中挑选出一条不同的信息。如果信息被随机发送，那么攻击者就必须发送另外n条信息以确保并发的信息不会混淆。

  对这个方法的一个防范措施就是转信器广播，转信器将把每一个并发的信息广播发送到其他的转信器组，以此来代替使用一对一的邮件传送来把随后信息发送到一个特殊的转信器中，而且只有一个转信器拥有解密下一个地址的正确密钥。
- **重复攻击** 攻击者在信息通过时抓取一个信息的拷贝，稍后再发送此信息，最终信件将沿着转信器链路一路前进直到到达和以前一样的目的地。如果攻击者跟踪发往全部目的地的所有邮件并重复操作几次，那么这样一个恒定的接收者就出现了。这就是所谓的目的地。

  对此最好的解决方法就是要求每个信息都包含有一个由发送者随机产生的独立ID数，转信器在一个很大的文件里存储这个ID，如果遇上另外一个具有同样ID地址的信息，那么转信器将丢弃这种信息。ID的尺寸应该足够大，以保证随机选择的两个ID几乎

不能相匹配。

- **伪造的邮件攻击** 伪造发送到SMTP（简单电子邮件传输协议）的邮件相对来说是比较容易的，当有些人发送含有非法信息的信件的时候，他们可能会假冒你。如果警察施压要求转信器操作者揭露出用户名，那么你将可能被控染指一些你根本没有做过的事情。

  许多转信器使用的密码就是对这种问题的一个简单的防御办法。匿名转信器不会发送那些本应该是从特定地点而来的邮件，除非这些邮件包含有密码。一个更完善的系统需要邮件用正确数字信号做标记。

政府可能直接处罚犯罪，事实上也是如此。但不能不分青红皂白只通过讲话、讲话内容以及与防范的危险没有任何必要关系的东西来间接地处罚犯罪。——摘自史蒂文斯审判员在美国俄亥俄州选举委员会上的主要意见。

每一种解决方法都是来自于David Chaum的一本描述一种叫做混合的过程的书[Cha81]。它的详细内容被用做目前在网上很流行的大多数完善的转信器的结构。Lance Cottrell编写了Mixmaster（混合控制），它是一种基于UNIX的程序，这种程序使用书中描写的更坚固的结构来发送匿名邮件包。

最主要的不同就在于地址信息的结构上。第一种转信器类型是在网络包络（译者：原文是neting envelopes）里对它们的数据打包分组。每一个沿着链路的转信器都打开一个包络并根据内容做正确的事，混合控制维持一个独立的地址块组，每一个地址块组都通过整个转信器链路。它看上去更像邮政局根据不同的接收者表格而发送杂志的一个分发表格，每一个接收者都在接收到他（她）的东西后便划掉自己的名字。

在这种形式里安排信息有两个优点。第一个是信息尺寸的压缩没有必要的原因。如果外层包络被去掉，那么信件的尺寸就会收缩。这可以用附加填充来补偿，但要填充到合适的尺寸可能是非常复杂的，这是因为密码器中不同的数据块可能具有不同的尺寸，如DES那样。第二个优点是减少了加密时间。加密块不一定要在转信器链路的每一个阶段都被加密或者解密，只有地址块才需要这样操作。

设想一个信息要经过5个中继，如果所有的加密都被删除，那么一个混和控制的标题就将包含一个像表10.1那样的表格。

表10.1 一条匿名信息的途径

转信器入口处	下一个目的地	数据包的ID	密钥
BoB	Ray	92394129	12030124
Ray	Lorraine	15125152	61261621
Lorraine	Carol	77782893	93432212
Carol	Gilda	12343324	41242219
Gilda	Final Location	91999201	93929441

加密被移除是为了展示工作过程。标题指明了邮件应该从Bob那里发送，在邮件到达最终位置之前它经过Ray、Lorraine、Carol和Gilda。ID数据包被用在每个转信器以防止重复攻击。

在混合控制中使用了两种类型的加密方法。第一种，每个标题的入口处用转信器的公共密钥加密，下一个目的地址、ID数据包以及Ray的密钥都被Ray的公共密钥所加密。只有每个转信器的正确接收者才能够解开入口的密码。

第二种加密方法使用表中储存的密钥，理解它的最好的方法就是使转信器所做的事情直观化。步骤如下：

1. 用它的密钥解码它的数据包。这揭示了一个目的地、ID和密钥。
2. 使用密钥解密它下面的每一个入口，混和控制技术使用三重DES编码信息。
3. 移动自身到表格的底部并且替换转信器名称、目的地信息和含有任意数据块的ID，这样就可以使路径隐蔽。

如果这些步骤被列表中的每个转信器成功重复，那么初始表格就将被正确地加密。标题的每个入口都必须用每一个标题上面的密钥加密，例如，Carol的入口看上去应该像这样：

$$E_{12030124}(E_{61261621}(E_{93432212}(PK_{Carol}(...))))$$

Bob转信器将除去由函数$E_{12030124}$指示的第一加密层，Ray的转信器将除去第二层，Lorraine的将除去第三层，最后剩下的数据块由Carol的公钥加密。

当标题到达了最终目的地时，每一块都以相反的顺序被重新加密。这形成了一个像有某种保证的字母信号链路，每一步都必须按顺序完成，而且每一步骤只能被持有与其相匹配密钥的人完成。最后的接收者可以保存这个标题，并检查过程是否正确无误。

链路的最后密钥，在这个例子里就是Gilda入口的那个密钥，是用来加密信息的。转信器没有必要在每一个步骤对信息进行解密。

如今的混和控制添加了一个具有20个标题入口的块到每个入口的最顶端，每块占512字节。如果信件只需要经过5个转信器，那么其他的将被填充随机噪声。表的每个入口都包含1位用来证实这是最后转发的位。如果该位被设置，那么密钥就被用来解密主题。

每条信息的主题都保持着相同的尺寸。如果当前的这条信息太短，那么就添加填充直到信息有20KB长为止。如果信息太长，那么它就会被分成20KB的数据块。尺寸是灵活的，但对所有的信息，它都应该被设为一个常量，这样可以防止任何人通过信息大小，或大小的改变来确认信息。

现在的Mixmaster软件能在www.obscura.com里找到。这个软件不能够随意出口，所以，你必须发送一条信息证明你遵守美国政府的出口法律。你可以在Lance Cottrell的主页上找到更多的信息，网址是http://obscura.com/loki/Welcome.html。

10.2.1 其他的转信器数据包

在UNIX系统中一个最好的、最丰富的程序就是Mailcrypt，它是用Emacs-LISP（编辑程序宏指令表处理语言）编写的为了和GNU设计的公众GNU Emacs程序共同使用的程序。这个软件则是由Patrick LoPresti开发，用来处理所有邮件的基本加密工作，包括加密输入邮件和解密输出邮件，还有评估信号。这个软件和主要的UNIX邮件相互作用来阅读像MH-E、VM以及Rmail这样的程序。

这个软件也包括了一个产生转信器链接的很好的工具。当选择这个选项的时候，它就会自动产生具有加密信封的嵌套数据包，这样就可以被由Raph Levien维持的表格里的转信

器所理解。

你可以为将来的用途而创建所有可能的转信器链接表。它可以是硬编码表，也可以有灵活性。可以具体指定，例如，每次Mailcrypt沿着链路发送信息的时候，它都应该选择一个不同的4个转信器的随机顺序。也可以要求Mailcrypt根据Raph Levien的表格使用4个最可靠的转信器。它在信息管理方面提供了充足的灵活性。要得到Mailcrypt，可以到以下这个网站：http://cag-www.lcs.mit.edu/mailcrypt。

Mailcrypt也使利用假名字变得很简单。可以为秘密本身创建一个PGP（一种加密工具软件）的密钥对，并公布它们的公钥。如果你的假名为Silence Dogood，你可以通过转信器的一个链接发送信息。最终的读者可以确定信息仅有可能是从Silence Dogood来的，因为他们可以恢复正确的公钥并确认水印。一些人试图假冒某人，但他们没有相匹配的密钥，所以，他们不能用这个假名发出任何信息。

PGP，*或者*Pretty Good Privacy，*是一个已知的比较好的加密软件包。*

另一个为了把所需要的转信器信息链接在一起而发展起来的程序是由Raph Levien编写的Premail（预发送邮件）。这个软件被设计用来取代发送邮件，发送邮件是一个基于UNIX系统只能提供低级别SMTP的软件。Permail可以通过修改自身规则采用同样的参数，包括附加的运用转信器链接的命令装置，你可以在你选择的任何地方用它来取代发送邮件。

Premail有几个主要的选项。如果你只在有接收者user_id的标题里简单地包括一行 Key：user_id，那么Premail程序就会在PGP文件中查找密钥，并且从外面用这个公钥加密文件。如果你的标题包括一行 Chain：Bob；Ray；Lorraine，那么Premail在文件发送到目的地之前，将重新安排以使得邮件的标题依次为：Bob的匿名转信器、Ray的匿名转信器和Lorraine的匿名转信器。如果喜欢添加 Anon-From：标题区域，那么也可以具体指定一个匿名回复邮件的地址。Premail是非常灵活的，因为它会从流行的转信器表格中随机选择一个转信器链接。只要在标题区域指定所希望的跳跃数，如：Chain：3，Premail就将会从Raph Levien转信器表格中选择出一个最好的转信器。

10.2.2 分离路径

现行的转信器集是非常简单的。一条信息通过一条路径发送出去，在沿线的每个步骤，转信器去掉输入的发送者的名字并加上一个新的匿名名字。因为匿名转信器将会用原始发送者的名字来置换匿名名字，所以回邮可以通过这条路径返回。

这种方法仍然遗漏了一条路径——尽管那是最强大的那一个路径。如果攻击者能够沿着链路损害转信器，那么他们当然可以找到一个方法来发现原始发送者，你所需要的只是知道链路上的最后一个名字，这个名字又是回邮链路的第一个名字。

一个较好的方法就是使用两条路径。输出邮件可以沿着一条路径被传送，这条路径不需保持对沿着其本身移动的邮件进行跟踪。回邮可以通过由原始发送方指定的另一条路径返回。例如，原始信息可以通过转信器anon@norecords.com，该转信器不会记录发送信息的人。接收者可以通过使用加密信件里的返回地址来发送返回邮件，这个地址可能是my-alias@freds.remailer.com。只有能够重新编码信息的人才可以攻击my-alias@freds.remailer.com以跟踪返回发送方的链路。

这种方法可以防范访问标题的某个人，而标题通常会给出匿名回邮地址。现在这个信息可以加密在正文当中。这个计划仍然是脆弱的，因为那些知道回邮地址my-alias@freds.remailer.com的人可以强迫Fred泄露你的名字。

另一种不同的解决方法是把返回地址分离成一个秘密。当你在freds.remailer.com开户头的时候，你可以填写你的回邮地址，比如R_1。这个地址并不将成为回邮地址，它仅仅是揭示你的回邮地址的秘密的一半。而另一半，R_2，将放在加密的信件实体里发送给你的朋友。如果他们需要回应，就要在返回信件标题里包含有R_2。这样，freds.remailer.com就可以把R_1和R_2合并起来并显示出真正的回邮地址。

回邮地址中发送者的那一半可以在任何时间到达匿名收件箱，发送者可以让它在那等待以便信件可以尽可能快地被改变路径，或者发送者可以在三天之后再发送这个地址，使在那等待的邮件恢复。

这种分离秘密可以用很多不同的方法产生。最简单的技术就是使用像第4章里描述的XOR（异或）加法。这是最简单的，最快的，又是最安全的，惟一的实践性困难就是把它转换成为合适的ASCII文字。电子邮件地址通常是一些英文字母和简单的标点，代替简单地产生一个与回邮地址进行异或的8位二进制编码，预料字母表里的偏移量可能更为简单。你可以创建一个有60个左右的字母的字母表，这些字母都在所有邮件地址中使用，并称之为字符串C。分离一个邮件地址由以下的以一个字符一个字符为基础的步骤组成：

1. 从C中选择一个新的字符，把它存储在R_1中。设x为该字符在C中位置。
2. 从邮件地址中编码一个字符，在C中找到这个字符的位置并且向下移动x字符到x位置。如果到了末尾，就重新开始。
3. 存储这个字符到R_2中。

它的相反过程也可以很容易地计算出来。这将产生一串特定的字符，把电子邮件地址二等分，成为R_1和R_2。R_1存储在一个匿名转信器中并附加了某个假名；R_2被发送到任何想给你回应的人。他们必须在他们的信件中包括R_2以便转信器能够为你组合回邮地址。

一个更完善的方法就是使用接收者的数字信号。这种交流的启动程序可以在回邮转信器上存储三样东西：假名、回邮地址的一半R_1以及可能被响应的人的公钥。当那个人回应时，他或她必须发送$f_c(R_2)$。这是被私钥编码的另一半秘密。转信器有与其相应的公钥以便它可以恢复R_2，并在R_2的路径上将信息发送出去。

这些系统可以是渐变的，不规则的。一个转信器必须保护自身不受别人用假名伪装来攻击它的大门，这可以通过用接收者的公钥加密转信器文件来实现。用个例子会比较好解释。想象一下，Bob要通过freds.remailer.com和Ray建立起匿名通信频道。通常，freds.remailer.com将会存储Bob的回邮地址，称之为B，并使之与Bob的假名maskT-AvEnGrr相对应。当然，别人可以查看这些文件来恢复回邮地址B。

freds.remailer.com可以通过建立会话密钥k_i来保护自己，并用Ray的公钥$f_{ray}(k_i)$加密它。这个数值和信息被发送给Ray，然后在丢弃k_i前用一些类似DES的加密算法使用k_i给B加密。现在，只有Ray才持有可以恢复k_i和形成B的私钥。freds.remailer.com被关掉了，即使它想揭示B，它现在也不能做了。

不够理想的是，这种解决方法只能对特殊不间断的通信进行处理。对于Bob要发送邮件的每一个人，建立一个不同的会话密钥是可能的。这样存有B备份文件的转信器，在下一回

收到maskT-AvEnGrr的信的时候，通过附加的会话密钥，从而增加了B被恢复的可能。

10.3 匿名网络

匿名网络传送单一的数据包，这样就出现了匿名Web页。这一点都不令人吃惊，程序员把这些想法延伸到提供在因特网上指明传送所有的数据包路线的绝佳工具。它们实质上就是创建一个TCP/IP代理，加密离开计算机的所有数据，然后弹回它并通过一个服务器网络最终传播到整个因特网。

每一个系统都提供大量的匿名措施来防范攻击者，对所有用户的数据形成了一个能有效地模糊每条路径上的始点和终点的暗影。然而，没有任何一种方法能完美地对抗无所不知和无所不能的攻击者，他们可以用自己的数据包探测并监视网络上所有的终端。每个系统都有一定的强度，但有些较为脆弱的地方就容易被上述情况所利用。

10.3.1 Freedom Network

Zero Knowledge Systems设计和建立了Freedom Network，一个由加密数据包的完善协议连接的服务器集合。这个网络持续到2001年，后来因为公司经济原因而被关闭了。这个网络在保护因特网秘密方面仍然是最有雄心的工具之一。

> Freedom Network是得益于Onion Routing Network的灵感，并在此基础上由Paul Syverson、Michael Reed和David Goldschlag在Naval Research Labs开发的[SRG00，STRL00，SGR97，RSG98]。

这个网络由一个的匿名因特网代理集合（AIP）所组成，当把数据转送到其他的代理时，这个AIP对信息进行解密和加密。如果一台计算机要建立一条通往因特网的路径，它可采用下列步骤：

1. 网络的中心是NISS（网络信息状况服务），一个中心电脑维持着一个操作AIP及其公钥的表格。
2. 中心电脑掌握这些设备的一个清单，并选择一个随机路径通过这一组设备。这个路径通过使用有关距离的信息来建立算法和加载优化程序。短些的链路提供较好的服务，而长些的链路对侦测提供更多的抵抗能力。链路在不同的国家运行还可能会提供一些额外的合法协议。
3. 中心电脑在链路的的AIP上使用Diffie-Hellman密钥交换来转让一个密钥。
4. 传出的信息数据依次经过每个密钥进行加密。如果f_k是使用密钥k的加密函数，那么$f_{k_1}(f_{k_2}(\ldots f_{k_n}(data)))$就被发送到链路。$K_i$就是链路上第$i$个AIP密钥。
5. 每个AIP接收到它的数据包并且在传输之前使用协议层密钥剥去最外层。
6. 链路上最后一个AIP发送数据包到正确的目的地。
7. 返回数据相反地经过相同的链路，每个AIP使用相同的层密钥加密数据。
8. 这个计算机去掉n个层。

Zero Knowledge参考这个过程为压缩嵌套加密，而实际的过程更为复杂和精密，使用这么多加密，又要提供足够的实现性不是一个容易的技巧。

10.3.2 PipeNet

PipeNet是一个匿名网络，由Wei Dai发明。它也是依赖于一个指定加密数据包的计算机网络，主要的不同就是协调数据包流的同步机制。在每个时钟步骤，网络上所有的计算机都接收到数据包，运行必要的加密过程，然后传送它。如果某个数据包没有到达，就没有可发送的包。

这个方法防止无所不知的攻击者通过查看数据流而发现谁和谁通信。在自由网络，一个很笨的用户可能会因为传输大量数据而泄露他的路径。一个无所不知的攻击者可能无法破解密码，但可以通过计算文件大小探测出它的目的地。理论上，一个大的用户数据库将会提供足够的覆盖层进行保护。

PipeNet发送信息的严密过程确保了机器之间的每一个条链路在每一步中只能携带相同数量的信息，数据通过链路就像士兵行军的方阵一样有严格的规定。

尽管如此，这个过程也有它自身的弱点。如果一个数据包被破坏或网络上的一个节点被关闭，整个链路就会中断。如果数据不返回，它也就不能发送（参看[BMS01]）。

10.3.3 Crowds

Crowds工具是由Micheal Reiter和Aviel Rubin发明的，它为网络浏览提供了一个简单的机制，就是和Freedom Net或PipeNet具有一样的匿名性，但并没有那么安全。然而，实现和运行它还是很简单的（参看[RR98]）。

这个协议非常简单。网络上的每台计算机都为一个网络文件接受URL（Uniform Resource Locator，在Internet的WWW服务程序上用于指定信息位置的表示方法）请求，它产生一个随机选择来满足请求或者向下传到另一个随机选择的用户。如果你需要查看一个文件，在你的请求最后被正确的服务所接纳并且通过链路返回之前，它可能要通过许多不同的人。

这个协议提供高级别的匿名性，但它还是不能欺骗一个查看所有站点的无所不知的攻击者。这个简单性提供了强大的混淆状态，你的机器可能接收到Alice的请求，但没有办法知道Alice是否真正对信息感兴趣。她的机器可能仅仅通过了来自别人的请求，而这个人又只是仅仅通过了来自另一个人的请求，而另一个人又只是通过了又另一个人的请求，如此等等。链路上的每一个个体只能知道某人对信息有兴趣，但是对于到底是谁，却是不能确定的。

10.3.4 Freenet

一个最成功的匿名公共系统就是Freenet——一个由Ian Clarke原始设计的对等网络。这个计划本身是在高度发展的，并且公开的代码资源分布可在下列网站找到：http://freenet.sourceforge.net（参看[CSWH00，Cla99]）。

这个系统通过一个随机服务集合来分布信息，这个服务器集合贡献它们备用的磁盘空间来查找公共文件。这个网络没有中心服务器，所以所有的信息搜索都分散在整个网络，每台机器记住一个先前搜索的确定数量，以便它能够应答公共文件的请求。

每个文件都通过3个不同的关键字而使网络能够辨识它，这些关键字有160位数字，由SHA混编功能产生。如果你需要检索一个文件，你的请求必须有3个密钥中的一个。搜索过程是稍微有些随机和杂乱的，但也能抵抗对网络的破坏。它有下列步骤：

SHA，也就是Secure Hash Algorithm，是一个像MD-5那样的加密安全的函数。

1. 通过具体指定密钥数值和一个“使用期限”的数来开始搜索，它限制了你想搜索的节点深度。
2. 选择网络上的一个节点开始搜索。
3. 这个节点检查这个密钥是否匹配本地存储的某些文件。如果匹配的话，节点就把文件返回。
4. 如果本地没有匹配的话，节点就对最近搜索的高速缓存进行检查。如果密钥在缓存中找到，节点就检索这个文件。在某些情况，文件已被本地存储。在另一些情况下，节点必须返回信息源进行检索。
5. 如果这里没有匹配，服务就会请求链路上别的匹配，并重复该过程。在每个步骤，“使用期限”计数器被被减小1。如果它减到0，那么就意味着搜索失败。

实践中，高速缓存的快速搜索基本上可以检索到大部分通用文件。

分配到每个文件的密钥可以由不3种不同的方法产生。作者开始先为文件指定一个标题*T*。这个字符串被转换为“关键字标记”关键字。这个值通过计算*SHA(T)*来混编，然后被用来加密并标志这个文件。使用*SHA(T)*来加密文件保证了只有知道T的人才能阅读文件。个体服务器可以保持一些被各自标题混编加密的文件数，但只有知道标题的人才能够阅读它们。这可以为那些从来不真正了解自己硬盘资源的人提供确定的否认本领的数量。

这个混编标题也被用来作为伪随机驱动公/私密钥产生程序的依据。当大部分公钥被选择为真正的随机信息源时，这个算法使用混编T以确定每个知道T的用户能产生相同的密钥对。这个公钥被用来标志文件，提供一些使标题和文件相匹配的保证。

这个机制离完美相差甚远。任何人都想得到相同的标题，攻击者可能故意选择一个相同的标题来替换这个文件。Freenet通过创建一个连接到作者的“标志子空间密钥”（译者：原文是signed subspace key）置人文件中来对抗攻击。这个密钥的创建更多的是包括：

1. 首先作者公布一个公钥限制他或她的身份。
2. 这个公钥和标题被独立地混编。
3. 异或所得结果并再次混编：*SHA(SHA(T)⊕SHA(public key))*。
4. 私钥联合公钥被用来标志文件。
5. 公布这个同时含有标志子空间关键字和签名的文件。

现在检索文件必须要求知道T和作者的公钥。然而，只有作者才知道私钥，所以只有作者才能产生正确的信号。

第3个密钥，恒定混编密钥，由混编整个文件而产生的。在公布它的时候，作者可以决定文件要包括哪些密钥。

明显地，维持一些中心密钥、文件和服务器的索引，会让每个人的生活更加简单，同样也包括那些审查系统的人。Freenet避免了这个过程，但也采用了一个步骤使搜索过程更为简单。当新文件被添加进网络，作者在服务器上用相类似的关键字置放它们。这个存储过程通过查找密钥来搜索网络，然后将文件置放在那里。

10.3.5 OceanStore

一个更有抱负的、持久的、坚固的、分布式的存储设计就是OceanStore，它是由加州大学伯克利分校的教员和学生集体开发的。这个设计的许多基本的想法和风格的灵感来自于Freenet，但是为了升级等原因，它还添加了许多改进的工具（参看[KBC+00]）。

一个最有意义的添加是确保文件被修改时不会被损坏的机制。新的数据块可以被插入到文件里，并且这些变化传播于整个网络直到所有的拷贝都是通用的为止。这个机制也包含一个更完善的路由结构来加速文件的标识和定位。但所有的这些细节已超过了本书的范围。

10.4 远景展望

在不久的将来，互联网上的每台机器都将是能够会发送和接收邮件的"优秀公民"。现行转信器的最好解决方案就是创建转信SMTP端口到一个匿名转信器的工具。在某些程度上，有一些已经做到了。它们接收到邮件信息并往下传送到最终目的地，这将是简洁的，例如，为Eudora或者别的邮件程序创建一个插件程序MIME模块，这样将会识别出MIME模式"X-Anon-To:"，然后立即把邮件重新发送出去。

在很大程度上，这些工具并不是最重要的步骤。这些工具只是用在转信器的所有者希望抵制揭示隐藏身份的时候才有用。

网络上的匿名日期服务也是很有必要的。尽管许多转信器披着"计算机朋克"（译者：cyberpunk）的外套，但是毫无疑问，转信器有许多合法的需求。多数主流的转信器可以做大量的商业工作，并帮助需要匿名通信的人们。

10.5 小结

- **伪装** 寄信人和接收者之间的路径被接收者通过中间机器删除回邮地址而隐藏起来，更完善的系统能够弄乱可以进出转信器窥视邮件信息的人的连接。
- **安全性** 基本匿名转信器只能与转信器链路中最强固的链接一样安全，如果运行转信器的人选择记录信息量，那么这个人就可以破解匿名。这可能会被司法执行机构通过搜查证和传票强制执行。

  试图隐藏信息量分析的、更完善的转信器是非常安全的。任何查看转信器网络的人只能对信息流出或流入网络做高层次的记录。尽管如此，还是很有可能跟踪信息流。系统对第11章中的Dining Cyptographers网络不能提供绝对的安全。

  数字混合必须被正确地构建。不能简单地使用RSA来标志信息本身，必须标志一个信息混编。[PP90]展示了如何利用它的弱点。
- **怎样使用** WWW网站是大多数人可利用的最简单的选择，更多的改进软件可以在Mixmaster档案和附录D列表上找到。

第11章 秘 密 广 播

漫谈话题

Chris: 我听说Bob被解雇了。

Leslie: 解雇？

Pat: 我听说原因是因为他的销售额下跌了。

Chris: 是的，他没有完成承诺的销售任务。

Leslie: 但是每个人的销售额都在下降。

Pat: 他说他有一个好的销售地域。

Chris: 好？他的销售额降得更多。

Leslie: 比什么降得多？

Pat: 这些销售额或许相对往年是下降了，但对别人来说是个丰收年。

Chris: 我想每个人都相对以前下降了。

Leslie: 可能有些什么别的原因。我听说他喝得太多了。

Pat: 我听说他受不了老板的愚蠢。

Chris: 实际上，他老板是有才气的。是Bob自己的问题。

Leslie: 这不合乎情理。

Pat: 好了，这讲得通。可能真正的原因就是我们所说的每一件事。你必须把它们总结出来。

11.1 秘密发送器

你怎样播送信息使每一个人都阅读它，但是却没有人知道信息的来源？对于发送无线电广播的地方，能通过简单的指向性天线轻易指出其位置。匿名重邮器能将返回消息来源的路径切断，但这些路径还是能被泄密或者追踪出来。事实上，所有网络上的信息都能被追踪，这是因为数据包都是从一个地方移动到另一个地方。这样做通常是不切实际的，但仍有可能性。

没有一种方法能提供绝对的安全，但是David Chaum发明的一种算法可以使追踪信息来源变得不大可能。他将这种算法称为“Dining Cyptographers”，他参考了一种著名的计算机系统设计问题“Dining Philosophers”。在Dining Philosophers问题里，n个哲学家围着餐桌，同时有n双筷子在桌上，使每个人都夹在两双筷子中间。吃饭时，每个人必须迅速抓起一双筷子，如果事先没有进行过协议和安排，将会有人根本吃不上饭。

Chaum将这个难题描述成为一个原则性问题：三个密码员在一起吃饭，其中有一个来自国家情报机构（NSA）。服务员过来告诉他们，三人之中已有一人计划好付账了，但他没有说明钱是谁付的。三个人为了这个问题起了争执。因为两位非政府人员都不希望接受这个来

白NSA的匿名善举。但是，因为他们都有匿名的需要，所以他们协定通过一个简单的投币算法来解决问题。这种算法不能使人知道是谁付的账，但他们可以知道付账的人是否来自NSA。

这个故事构想与位有点牵强，但是仍然有效果。总的来说，一个成员将发送一位信息到桌上，每个人将得到相同的信息，但无人分辨得出发送信息的人是谁。也有许多其他情况让他们陷于相类似的问题之中。举个例子，父亲回到家中发现后边的玻璃窗被击得粉碎，他怀疑到他的三个儿子，当然也有可能是小偷干的。他知道没有人会承认，在叫警察和报案之前，他运用了Dining Cyptographers协议的方法让三个儿子之一承认了是自己打碎了玻璃窗而不用自愿受罚[①]。

如果每一位信息能通过这样的算法，那么将没有理由为什么较长的信息不能通过同样的信道。有个问题就是无人知晓其他人将何时发布信息，因为没有人知道是谁在发布。最好的解决方法就是不去干涉别人，当一个空闲的时间段出现后，参与者要在开始之前等待或长或短的时间。开始传播时，要注意与此同时产生的错误信息。一旦此类情况发生，他们又将在重新开始之前等上一段时间。

这套系统同样可以简单地扩展出一个建立两人交流信息的方法，同时无人得知信息来源。如果没有人能准确描述出Dining Cyptographers协议信息的来源，也就没有人知道是谁收到了信息。如果发送者用饭桌上两人共同拥有的密钥将通信密码化，那么只有预期的接收者才能解开密码。其他人只能看到噪声，但没有人会看到消息往返的路径。

Dining Cyptographers协议系统浅显易懂。在Chaum最初的例子中，有三个密码员，每个人抛币并让他右边的人看到结果。这样，每个人就可以看到两枚硬币，由此判断他们的结果是否相同，再将结果公布出来。如果三者之一试图发送一条信息——在这种情况下NSA付了饭钱——然后他们将通过相同或不同的结果来交流答案。假如不同的结果用奇数代表"是"、"不是"或"NSA付帐"这样的信息将被传播。如果是偶数，就没有信息发送出去。

即使只有3枚硬币，由于它们都要与邻近的相互比较，也会显出复杂化。用一个例子来解决这个问题或许是最好的。表11.1列出投币得出的几种不同结果。每列显示出一个参与者投币的结果以及与他右边投币结果的比较。"H"代表正面，"T"代表反面。参与者1在参与者2的右边，参与者1可从列1比较两人投币的结果。在这个列中他可以报出结果是否一致。参与者2在参与者3右边，参与者3又在参与者1右边。

表11.1 三个哲学家送出的秘密消息

参与者1		参与者2		参与者3		
投币结果	是否匹配	投币结果	是否匹配	投币结果	是否匹配	消息
H	Y	H	Y	H	Y	否
T	N	H	Y	H	N	否
T	Y	H	Y	H	N	是
T	N	H	N	H	N	是
T	N	H	N	T	Y	否
T	Y	H	N	T	Y	是

①这可能是进步父母的做法，但不推荐你在家使用，应让你的孩子不要学习撒谎。

在第一行比较的情况中，有三对匹配且结果一致。0是偶数，无信息发出。但“无信息”可以看成与“0”或“关”等同。第二行中，有两种不同，是偶数，也无信息发送。第三行中，一条信息出现了，那就是一个“1”和“开”被通过。同样的情况出现在第四和第六行。

不需要把参与人数限制在三个人，多少人都是可以的，系统同样可以解决。每次投币结果会使最后的数目翻倍，一次是投币者，另一次是他的邻座。如果有个人改变了答案，相异的结果的总数只会是奇数。

两人同时发布信息会产生什么样的结果？由于两个变化相互抵消，那么协议将会失败。相异的总数将再次结束。如果三人同时发送信息，由于变化数目为奇数，协议就成功了。如果协议失败，应用者可以追溯信息的来源。你试着广播一位信息，但每个人计算的追踪结果是缺少一位信息。如果在再次开始前，每个人尝试发送停止信息并等待一个随机数字，那么奇数将不会再次冲突。

这个系统是否绝对的安全？假设你是其中一个桌边的人，每个人都在抛币，很明显会有信息出现。如果你不将信息发送出去，你还能确定是谁吗？假如你抛出的是“正面”，请看表11.2可能出现的结果。

我们从来不关心Slothrop是不是Slothrop。——Thomas Pynchon, Gravity's Rainbow。

方框里有四种可能出现的情况，每一种情况中，你都抛出“正面”。你可以看坐在你右边的人抛出的硬币。每次出现的相异数目都是奇数。因此有人发出了信息，可你能说出他是谁吗？

表11.2 系统是否能被破坏

你		参与者2		参与者3	
投币结果	是否匹配	投币结果	是否匹配	投币结果	是否匹配
H	Y	H	N	?	Y
H	Y	H	N	H	Y
H	Y	H	N	T	Y
H	Y	H	Y	?	N
H	Y	H	Y	T	N
H	Y	H	Y	H	N
H	N	T	Y	?	Y
H	N	T	Y	H	Y
H	N	T	Y	T	Y
H	N	T	N	?	N
H	N	T	N	T	Y
H	N	T	N	H	N

在表11.2中每个组合的第一条记录都有标记着第三个参与者弹出硬币的疑问，你不知道他将会抛出什么币面。第一条说明指出如果你隐藏的币面是“正面”，那么第二个人就是在

说谎并发送了信息。如果币面是“反面”，撒了谎和发送信息的就是第三个人。每行显示信息的字都是斜体的。

只要你不知道第三个人的币面，你就不能判断到底是另外两个人中的哪一个发送了信息。如果抛币很公正，你就不可能知道结果。这种情况同样适用于在系统外偷听的人。如果他们不是亲眼看到币面，他们就不会知道谁发送了信息。

Dining Cyptographers网络成员破坏这种通信有几种方法。假如几个人同时行动，他们就能比较邻近的币面和判断出信息发送者。如果桌旁的人是轮流宣布他们比较的结果，最后一个人就能轻易地通过改变他们的答案来改变信息。就是说，最后一个发言者可以判断出答案会是什么，这就是为什么要让人们同时公布他们的答案。

Dining Cyptographers系统给每个人提供发送信息的机会，而且可以不用公布他们的判断结果。这就像隐藏匿名转信器一样，不能简单地通过信息来源的路径而泄露身份。不幸的是，在互联网上不能随意地利用这套系统。也许更多需求出现才会更加普遍。

11.2 创建一个DC网

Dining Cyptographers（简称DC）方案很容易描述，因为在计算机网络上的方案因为工具的难度都省略了图片。在“桌边”，人人都可以同步公布他们的选择。参与者抛币，用菜单向他们的邻居公布选择以达到掩盖结果的目的。这两种方法对于实际执行都是重要的。

首要问题就是通过电脑网络抛币。显然，一个人可以抛币，然后隐瞒结果。最简单的解决方法就是运用单一路径检索功能。例如MD-5，或者Sneferu。

> Manuel Blum在[Blu82]中描述了如何在网络中进行抛币。这是建立一次性便笺或密钥的一个好方法。

电话簿就是一种良好的、可行的单一途径检索，只是不太安全，很容易将一个名字和一串数字相联系。但要利用普通电话簿将数字转换成名字就难了，这个功能不是太可靠，因为围绕这个问题还有别的情况。举例来说，你可以轻易拨通号码询问对方的身份，或者通过相反的号码簿或者能提供查看姓名的电话CD-ROM。

利用远距离抛币的单一途径检索功能是很简单的：

1. 你选择一个随机数x，然后发送$h(x)$给我，h代表可用计算机计算但不容易可逆的单一途径搜索函数。
2. 我不能从$h(x)$中算出发送过来的x，我只能猜到x是奇数还是偶数。猜测结果会传回给你。
3. 如果我的猜测正确，那么抛币将是背面。如果我错了，那么就是正面。由你确定是正面还是反面并发送x给我。
4. 我计算$h(x)$判断你是否在说谎。除非你能轻易找到两个数，x是奇数且$h(x) = h(y)$，否则，你不可能作弊。没有人知道在一个好的单一途径检索函数上如何做到这一步。

> 如果我提供x的前n位，这个协议可以变得更加坚固。一个预先计算好x和y的设置是不能用的。

这是一种邻近两个人可以不在同一张桌子抛硬币的算法。如果你在意曾为一部电影和朋友发生的争执，就可以通过使用这种算法利用电话簿进行单一途径检索函数计算。抛币将是公正的。

第二点要注意的就是无论抛币结果是否相同，都应该同时公布答案。Chaum的文章建议要同步公布结果，但不要同频率，这要求有比现有网络更精密的电子设备。更好的方法是要求人们通过行为议定后再公布答案。

解决方法是简单的，首先整个小组对存储的短语和收集的信息取得一致意见，这种做法是越迟达成越好，这样可以防止有人利用算法提前掌握运行规则。用B作为随机数字的函数式。公布他们的回答，n个参与者按照以下步骤行事：

1. 秘密抽取n个随机函数，$\{k_1, \ldots, k_n\}$
2. 分别算出数值，把B放在答案前，将各输出函数和其函数值译成密码。$f_{k_i}(Ba_i)$是函数式，f是密码，k_i是函数，a_i就是要公布的函数值。
3. 将密码公布到整个小组，不需要注意顺序。
4. 当所有人都收到其他人发来的信息时，每个人就将k_i公布给整个小组。
5. 所有人将信息解密，查看，确信B是否在每组信息的开头。最后合在一起把信息解密出来。

这些约束协议使人撒谎的机会很小。如果译成密码的函数值中B不打头，一个精明的使用者就有可能发现两个不同的密钥得出不同的答案，如果他想告诉小组成员他要揭露一个匹配值，他就可以出示一个密钥。如果他想揭示另一个，他就可以发送另一个密钥。如果加密信息很短，这是有可能出现的。但如果加密信息都用同样的B开头，这种可能性就几乎为零了。找出这样相配的B几乎是不可能的。这就是为什么B在实践时要迟些达成。

这两种系统结合起来，可以使Dining Cyptographers网络使用异步通信变得简单。人们不需要同步公布答案，同样也不需要人们在同一地点抛币了。

11.2.1 欺骗DC网络

有许多各种各样的方法可以破坏DC网络，但也有许多相应的防卫措施。如果有人共谋向桌边的其他人泄露他们的位信息，那么对于停止追踪将无计可施。在这些情况下，匿名转信器会比较安全，因为这和他们之间的最强固的链接一样安全。

另一个主要的问题是阻塞。网络上的人随时可以发布额外的信息，从而干扰了合法信息的正常发送。例如，一条出自网络的信息，组员中有人心怀不轨，同时开始发送并破坏了其他的信息传送。不幸的是，DC网本身是将这个人的身份隐藏的。

如果社会问题变得重要，那么通过抛币方式还是可以知道是谁在干扰网络。这样做时，判断谁在发送信息是可能的。假定有针对反对广播的规定，当其他人使用DC网络发送信息时，就可以揭开传输链路，还是可以知道是谁在干扰。

如果每个人都上传一个“抛币”的关键数字签名，网络上的每个人都取得两个函数，自己的和邻居的，最佳解决方法就是在“抛币”上签上自己的或邻居的数字签名，迫使每个人都这样做了，以防止人们对自己的“抛币”作弊并更改签名。

如果只有一个人在使用DC网络来激怒大多数的人，比如也许是传送非法的威胁，这样的追踪方法是很有用的。如果一个人试图阻塞另一个人的通信，它就会暴露两个发送者。惟

一的办法是规定要合法地产生规则，在成员开始广播时作出判断，第一个传播者是合法的，其后的人都可能是阻塞者。

11.3 小结

Dining Cyptographers网络为针对通信分析提供的绝对安全创造了机会。没有人可以探测广播，当网络节点保持它们的“抛币”秘密时，信息被加密后没有人可以算得出接收者。

DC网络主要的局限就在于信息传输的高耗费。在广播这些信息的时候，网络上的所有成员都必须与他们的邻居“抛币”。它可以整批执行，但它仍然有一个重要的代价：*n*个人导致了网络带宽要增加2*n*倍。

- **伪装** DC网络提供了一个隐藏传输信息源的理想方法。如果这些传送被加密，那么只有指定的接收者才能浏览。
- **安全性** 如果抛币的所有信息都是保密的，那么系统是安全的。换句话说，如果所有的这些信息被泄露的话，这个组就可以追踪出发送者的路径。
- **怎样使用** Dining Cyptographers网络软件是为了让人们能够访问UNIX工作站和整个因特网的信息数据。

第12章 密 钥

主要观点

曾有来自国际科学基金会的一个新指令，它要求所有的研究员停止寻找“密钥”，因为它会大量解开他们研究中的秘密。这个要求是在一项新研究成果表明“整体观”比锁和钥匙更重要的比喻之后迅速被提出的。国家科学基金会的管理人员预测，这项新的指令会使发明提高47%，创造出重要的经济成果。

锁和钥匙这一比喻失败的最近消息，使许多研究人员大为震惊。兰登·摩菲，哈佛妇幼医院的一个癌症专家说：“我们正致力寻找一把钥匙，它会打开这个领域并为我们提供自然界所有丰富多彩的秘密。一把简单的钥匙就能为你服务。”

今后，所有的研究人员将训练自己的思维去寻找一张包含他们所有知识的全局图。人们认为内在的理解会在更快的时间段内产生更多的发明创造，因为重要观点（全局观）能见到整个森林，而不仅是一些树木。

“假设你站在山顶上，你看到的景色会比站在山脚看到的远得多，即使你有山下密道的钥匙也不行。”布鲁斯·康斯坦丁，一名国家基金会行政管理助手如是说，“我们希望我们的研究人员重点建立全局观。”

一些科学家对这个新指令犹豫不决，他们用自己的观点来驳斥它，“当然你能看到数里之外的景色，但是你却不能看清细微的事物。”匹斯堡大学的一名病毒学家马丁说，“我们之中的一些人需要重点关注小事物和细节，这样才能取得进步，这是关键。”

康斯坦丁打破这些异议，并建议病毒学家们避免狭隘的偏见才可能取得更大的进步。

“关键的洞察力，或者应该说是真正的观点，是视野狭隘的科学家避免看一个大的图片。我们想要更大的图片。如果那意味着抛弃拥有一个钥匙的希望，那么它只能这样。”

12.1 延伸控制

大多数隐写术竞赛都能找到一套能把一组数据看成是另一组数据的算法。在一些例子里，伪装数据是不够的，更强大的攻击要求更有力的措施，最有效的方法之一就是给算法加一些密钥位。

这在某种意义上是一个相对小的位集合，但是它却在算法中发挥着重要作用。如果没有正确的密钥，就不能解开算法的特定特征。这个密钥里的信息位在某种程度上说对操作数据是至关重要的。

隐写术里使用的大多数密钥技术是在基础密码学中使用的解决方法的延伸。一些基本的类型如下：

- **秘密钥匙** 一个被隐藏信息的钥匙，同样的钥匙必须可利用来揭示信息。这通常被称做对称密钥或私钥隐写术。本书没有使用第二个术语“私钥”，以便避免与公共

密钥方法中的术语相混淆。

- **公共机制或公共密钥** 一个隐藏信息的钥匙和一个解读信息的钥匙，即私钥和公钥，它们是不同的。这种方法常用于水印信息，因为某些人可以不用隐藏新的信息就能解读原信息——那就是说，不需要用到某人的新水印的备份。

  这里描述的算法中，没有一个能用最好的公共密钥加密系统的简单性来提供一个解决方法。它们通常依赖于难题或一些朦胧的问题去支持“私钥“的某一种形式。在很多情况下，根本没有真正的私钥。公钥仅仅被用于确认信号。
- **零位知识证据** 这些程序允许信息隐藏者隐藏信息，这样不需要揭示信息本身就可知道信息的存在。这个技术对于制作水印也是有用的，因为信息隐藏者不需要赋予别人隐藏相同信息的权力就可以证明他们隐藏了信息，也就是说，用相同的水印备份。
- **冲突控制码** 假设你有一个文件的几个备份，这些编码以模糊藏在里面的信息的方式防止你混合这些备份。例如，对于水印信息的一些基本攻击将被均衡到几个不同的备份，这些方法能够抵御这样的攻击。

有很多不同的方法用来执行属于这些类型的运算法则，最简单的就是在数据被隐写术算法处理之前对数据使用基本密钥和公钥算法进行加密。这个加密算法和隐藏算法是相对独立和互相区分的。

这个方法在本书的很多地方被提及，并且本书大力推荐这种方法。加密算法很自然地使数据看上去更加随机，而随机数据是隐写术的最重要的组成部分。因为加密是一个最简单的增加数据随机性的方法，所以即使秘密不是很必要，数据加密也还是很有意义的。当然，这个方法会偶尔产生过于随机的数据，这是一个在第17章被深入讨论的问题。

更进一步地看，保持加密与隐藏的相互独立仅是头脑简单的人的一种想法，好的加密算法可以作为条件指令与隐写算法相混合或匹配，我们没有必要放弃这种好方法。

但是由于隐藏算法对每一个人来说都是同样的（没有区别），所以把两个过程分开是有限制的。因为算法实质上是公开的，任何人都可以揭示隐藏的位。以前任何人使用这个算法解读信息必须在某一特定场合下，而现在他们则可以在其他的场合下重复地使用同样的算法了。因为加密系统是非常强大的，所以他们对位信息无可奈何。但他们可以相对容易地找到信息位并用他们自己的位来代替它们。

对隐藏过程用密钥加密，保证了只有持有正确钥匙的人才能隐藏或解读位，这是通过在过程中增加额外的复杂层来限制攻击者。

本书的很多基本算法，是使用简单的钥匙来控制一个算法的随机选择。在决定这么做了以后，用一个钥匙开启一个加密安全随机数字生成器，就是给这个方法加上一个钥匙的最简单的机制。

第9章的算法是通过选择一个元素的子集在图像和声音文件的最小有效位里隐藏信息，这个选择过程是通过一个随机数字生成器重复地混编一个钥匙来产生的。在第13章里，用来计算数据元素分类清单的函数也包括了一个钥匙，如果同样的随机数字流量是不可利用的，那么位就不能被摄取。

更完善的系统把密钥合成一体，即使钥匙是处在算法的深层。一些密钥试图约束难题答案被解除的方式，另一些则限制它在数据中被编码。

很多这些更新的先进系统展示了任何一个计算过程是怎样被歪曲成一些额外的位的，大多数算法都包括一些关于位置、顺序和过程的任意决定，并且这些决定可以由某些密钥来启动。作者对这个问题有足够的理解，提供了一些实际的讨论，使人们相信如果没有密钥，则解密过程是很难进行的。

12.2 记号算法

很多加钥算法通过扮演为文件的数字信号来提供关于文件真实性的某种保证，在数字信号能为任意文件提供某种确定性的所有情况下，这些解决方法都是非常有用的，对水印学来说尤其有用。理想的算法允许文件生成器用这种方法嵌入一个水印，而只有正确密钥的持有者才能够产生这个水印。

基本算法包括把图像和声音文件分成两个部分。第一部分包含在隐写术中不变的细节，如果信息被隐藏在最小有效位，那么这部分就是其他的位。第二部分是可被改变以隐藏信息的位，这个分组可以由一个密钥来定义。

一个文件的数字信号可以通过混编文件中不可改变的部分来建立，用一个传统的数字信号函数给这个数字做标记，然后在被留做隐藏信息的第二部分里给信息编码。数字信号可以用像RSA这样的传统的公钥算法来计算，或者可以使用密钥算法甚至是混编函数这样的更简单的算法来计算[Won98，Wal95]。

这个过程既可以使用从同一组位里获取的两个密钥，也可以不使用从同一组位里获取的两个密钥。第一种是用来定义会被改变的文件部分，它可能是一个用像素来隐藏信息的随机数字流量。第二个密钥实际上建立了这个信号。

这个方法对编码来说简单、直接和容易，但它是在文件最易受影响的部分里隐藏信息的，压缩算法和其他的水印可以通过改变这个部分里的数据来毁坏信息[CM97]。

> J. Fridrich和Miroslav Goljan建议在一个图像副本里进行自嵌套。一个块的内容通过图像嵌入到另一个块。修改和篡改都是可以逆向的[FG99]。

这一机制也可以通过把文件分成多个部分而被延伸。第i部分的信号可以被嵌入到$i+1$部分。图12.1展示了一个文件是怎样被分成5个部分的，一个部分的数据在下一个部分里被隐藏。

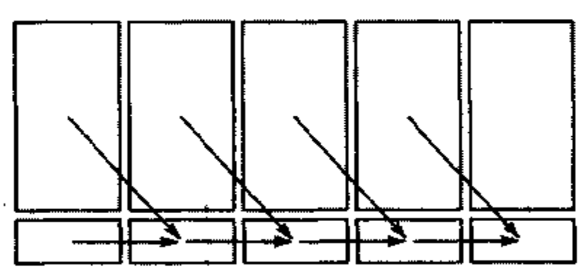


图12.1　一个嵌入数字信号可以被编入一个文件。在这个形象的比喻里，文件被分成5个大块。每一个大块又被分成两个部分：一个含有不可改变的部分，另一个则用来隐藏信息。从部分i而来的信息被用来产生一个嵌入在$i+1$这个部分的数字信号

12.3 公钥算法

很多研究者正在为隐藏信息而试图发展公钥算法，它能提供和公钥密码术一样的很多特征。这些系统允许用户用任何人都不能恢复信息的这样一个方法来隐藏信息，但是理想上还是应该用任何人都不能复制信息的方法。这个特征是水印学所希望的，因为它对一般用户（或者是用户计算机）检查信息水印是很理想的。

所有的算法都是新的，并且未经试验，但是它们仍然为水印文件提供了令人兴奋的可靠性。如果它们能被发展到足够强大以抵制攻击，人们就将可以使用它们追踪版权和保证文件的真实性。

12.3.1 约束性的难题

有一种策略是使用很难解决但是又像公钥基础那样很容易检验的问题。怎样解决难题的知识就像一个私钥一样，怎样校验难题就像一个公钥一样。真正的挑战是找到在正确的途径里运行的问题。

NP-complete问题的类型包括像布尔数学体系的可满足性、“旅行推销员”问题以及很多其他的问题这样的计算机科学挑战类型[GJ97]，它们通常很难解决但是却很容易确认。不幸的是，用适当的强度混合并不总是能确认一个特殊的问题。

这个方法现在还没有取得具有历史意义的成功。一个早期公钥系统是由**Ralph Merkle**和**Martin Hellman**使用以渐缩算法而著称的**NP-complete**的问题而产生的（给出n个项{w_1, ...w_n}的加权值，找到一个加权值为w磅的子集）。他们的算法产生了习惯的渐缩，这种渐缩显现在提供一个它们很难被正确地打包的担保。只有有一个秘密值即私钥的人，才能更容易地决定目标的正确子集以便放入渐缩算法。在实践中，有一些人破坏了这个系统。

另一种技术就是避免在系统中插入一个陷门或私钥。**NP-complete**问题的固有强度是很有威慑力的，这意味着那个想做一些标记的人必须具有解决**NP-complete**问题的能力。

在一种方法里，一位名为**Gang Qu**的研究者设想产生这些解决方法的能力是被一个水印保护着的。一个知道怎样用最佳的方法解决“旅行推销员”问题的人，可以把方法销售到航空系统、商业、或是旅行推销行业。为了保护他们的习惯路线的版权，这些客户在文件里隐藏了一些信息以证明这些方法是他们的。一个竞争的商业可以发展它们自己的解决方法，但是任何人都可以通过找到这些被隐藏的信息查到起源。

这种技术通过强制把特定的参数设为特定的数值而把信息隐藏在难题的解法中[**Qu01**, **KQP01**]。任何人都可以检查这个方法和参数，但是因为问题推测起来很困难，所以要归纳成一个新的解决方法应该也是很难的。这种方法并不是真正地需要一组被标为“公钥”的位，但是它仍然表现在大多数同样的方法中。

信息通过在解决方法中引入新的约束被隐藏到问题的解法中。例如，经典布尔数学体系的可满足性问题，取n个布尔数学体系可变量，{x_1, ..., x_n}，并且试图给它们赋予真的或假的数值以便一组中所有的布尔数学体系项都是真值。通过把特殊值强制赋予一些可变量，信息就能够在答案中被编码，如$x_{14} = T$，$x_{25} = F$，$x_{59} = T$和$x_{77} = T$可以编码为字符串10011。

当然，加入一个额外的需要可能使一个解决方法变得不能运行，但是这最起码给设计者提供了一个机会。

这种技术可以被应用到很多NP-complete问题和其他问题上。很多问题都有复合的解决方法，并且信息能通过选择合适的编码方法被隐藏。图12.2是一个被上色的曲线图，几个节点选取固定的颜色。编码信息的一种方法就是抓取成对的节点并把它们涂上相同或不同的颜色，通过在算法寻找解答的时候把两个节点结合在一起，它们就能被强制拥有同样的颜色。而通过加上一个连接两个节点的棱边，它们就可以有不同的颜色。其他的问题通常都有一些活动余地。

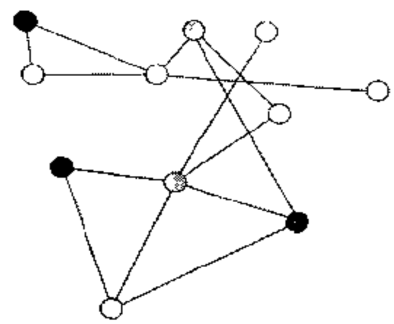


图12.2 这是一幅没有两个相邻节点具有相同颜色的曲线图，信息能被隐藏在强制某些特定节点具有特定颜色的解法中

沿着这个途径混编步骤，处理过程就可以被加强。设K_0为一个发生器的原始水印信息。它可以是发生器的名字，可以是一个电子邮件地址，或者是一个著名发生器的大表格的参考基准。设H是一个像MD-5或SHA这样的强混编函数。设$\{C_1, \ldots, C_n\}$为问题的一组约束，它对旋转移动以保存信息是公开的。这些在一个布尔数学体系的可满足性问题里的额外可变量可被设为真值或假值，或者它们可以是一个曲线图中的节点，这些节点能持有不同的数值。为了简单化，假设每一个约束可编码1位，下面的循环将可以编码水印：

1. 设$K_i = H(K_{i-1}, C_{i-1})$。被编码为$C_{i-1}$信息和有关$C_{i-1}$本身结构的一些数据都应该馈送到混编函数中，这信息可以包括节点数、可变量的名字或是布尔数学体系公式（设C_0零字符串）。
2. 从K_i中取出一位并把它编码为C_i。
3. 对所有n个约束项重复上一个步骤。

水印可以通过重复这个过程而得到测试。任何怀疑的询问者都可以从发生器中得到Ko并逐步执行这个过程以检查每一点的位。

这个方法可以用来保护复杂的工程设计，在这个设计中发生器必须解决一个难题。例如芯片设计师，当选择怎样布置晶体管的时候，他们通常要解决巨大的NP-complete问题。这个技术允许他们在芯片设计里编码一个水印，如*Copyright 2001 Bob Chiphead*中所述。

这个技术可以应用到更多的信息隐藏问题或水印问题上，但是它有一定的局限性。发生器必须有找到难题解决方法的能力——这些解决方法对一般的人来说是很难获得的。完成它的最简单的方法是制造一个仅仅能找到又大又难的问题的解决方法。只有计算机的拥有者

（或其中一种相似的力量）才能够产生这些方法。Ron Rivest和Adi Shamir论述用类似的解法创造小的标记。

12.3.2　使用矩阵式乘法

从一个文件中恢复信息需要找到一个扩大信息和将使伪装的数据最小化。一个解决方法是依靠相对随机的图像或声音信息，并设计一个能除去相对随机信息的恢复函数，而除去后剩余的部分就将保存着考虑范围内的信息。

Joachim J. Eggers，Jonathan K. Su和Bernd Girod建议用一种结构装置，在该装置中，矩阵性乘法会阻止隐藏数据但可保持与众不同的水印未受影响[ESG00b，ESG00a]。他们的方案建立于固有向量之上，固有向量就是在矩阵式乘法之后剩余的指向同一方向的向量。也就是说，如果M为一个矩阵，w是一个固有向量，那么在$Mw=\lambda w$里标量λ即为特征值。每个固有向量都有一个相应的特征值。矩阵式乘法能改变绝大多数的向量，但对于固有向量并不适用。为了简单，我们假设这些固有向量为一个单位长度，即$w^hw=1$。

探索此矩阵式乘法的特性需要假设文件的数据是相对随机的，并且平均值为0，声音文件能符合这种模式，但这并不适用于图像文件，因为图像文件的字节范围是在0～225之间。任何文件都可以被转换成一个平均值为0的文件，只要通过计算平均值，并将其从各项中减去。这个结果对这种算法来说可能并不够随机，但我们不妨假设它成立。

假设x是一个隐藏水印的数据向量，并假设它从平均值为0开始而且足够随机。什么是足够随机？也许通过描述我们想要的效果将能更好地去定义它。理想上，我们想要$X^hMx=0$或者至少足够接近于0，那么它将不参与计算而只有水印被留下来。

设w是某个矩阵M的固定向量，设λ为其相应的特征值。$Mw=\lambda w$。这个向量将被用做水印并且用一个β被加到伪装数据x上，理想上，β被选择以便使$X+\beta w$差不多等同于X，水印也可以被取出。

通过计算

$$(x+\beta w)^hM(x+\beta w)=x^hMx+x^hM\beta w+\beta wMx+\beta^2wMw$$

水印就可以从数据中取得。如果假设x为变量，前三项将接近于0，那么上式就变为$\beta^2\lambda(W_w^h)=\beta^2\lambda$。

如果M、β和λ的值被分配，那么一个公钥系统就可以被建立起来。为了存在或不存在的水印，任何人都可以通过计算y^hMy和断定它是否与$\beta^2\lambda$相符来检验一个文件y。

不过，这个方法还是有很多局限性。首先，x和w中元素数量必须相对较大。Eggers、Su和Girod用大约10 000和100 000的长度来报道结果，更大的值会帮助保证将x^hMx推向0的随机性。

其次，对任何攻击者来说，用信息发生器找到固有向量M是很简单的。一个解决方案就是选择有很多不同的固有向量$\{w_1,\ \ldots,\ w_n\}$的M，这些固有向量都是从同样的特征值λ开始。攻击者可以确认所有这些固定向量，但是通过相继减去不同的值w_i来移除水印就会被看成是残暴的强制性攻击。

当然，寻找固定向量的简易性，意味着某些人可以通过选择任意一个从特征值λ开始的

固有向量w_i来插人一个伪装水印。这也意味着这个算法不能像很多经典公钥算法那样被用来产生数字信号，但是它对产生阻止复制拷贝的水印仍然是有用的，一个盗版者将发现加一个指令复制被禁止的信号是没有用的。

更完备的攻击是有可能的。Eggers，Su和Girod 编辑了Teddy Furon，这样就可以确认一个跟踪错误的方法，并且用这个工具通过改变由长度为λ的固有向量{w_i, ..., w_n}定义的空间中信号部分的规模来移除水印。

这个算法可以通过选择不同的数值M来调整。因为一个n×n的矩阵能储存$n-1$个数值，所以Eggers、Su和Girod特别喜欢置换矩阵。一个置换矩阵的“乘法”也是相对简单的。因为在图像和声音处理中计算是经常用到的，所以使用设计的矩阵来计算离散余弦变换也是一个很好的选择。

这种解决方法并不完美，它的安全性也不高。但仍不失为将一个函数设计用于在隐藏信息不断扩大时，将掩饰性信息减至最低程度的良好范例。其计算过程也被加密以便M值必须被提交选取隐藏信息。

12.3.3 删除部分

本书中的许多算法是通过大量改变在文件中的许多不同的位置，然后求这些变化的平均值以找出信号来隐藏信息的。例如前一节所述的算法，可以从水印固有向量中加成百上千个小的数值到文件里。第14章中的spread-spectnim-like技术也是把信号传播到一个声音文件的许多不同的像素和单元，而这个信号是通过计算它们的加权平均值来获得的。

信息提取器并不需要求出所有选定位置的平均数以提取一个信号。这一发现归功于Frank Hartung和Bernd Girod[HG]。这些算法已经包括了一定数量的冗余以防备针对文件的恶意的或偶然的修改，如果这些算法被设计用来携带正确的数据，即使在任意一个元素数量面对改变的时候，为什么不安排接收者完全忽略这些元素的任意数量呢？

思考下列算法：

1. 产生n个“密钥”，{k_1, ..., k_n}。
2. 使用加密的安全随机数字生成器，用这些密钥确认文件中n个不同的m元素组。
3. 调整隐藏算法使它用信息能被恢复（即使有$n-1$个元素不可用或被损坏也能被恢复）的方法在mn元素里隐藏信息。
4. 隐藏信息。
5. 给n个不同的、有读取信息理由的人分配n个“密钥”。因为信息是用只有一个子集必需的冗余来隐藏的，所以这个算法仍然可以工作。

Hartung和Girod为{k_1, ..., k_n}n个数值使用术语“公钥”，虽然它们并不像很多传统公共密钥那样运转，这些密钥提供了大量的防伪保护，任何人拥有一个k_i就可以获取信息，但是他或她不能嵌人新的信息。如果k_i值的持有者试图嵌人一个新的信号，它将可能对其他$n-1$个密钥的持有者是不可见的，因为他们的密钥定义的是不同的子集。k_i的持有者能够改变尽可能多的元素，但是这不会改变由其他密钥持有者读取出来的信息。

当然，这个方法也有其局限性。k_i的值并不是真正公开的，因为它们不能够无限制地循环。它们也可以被用来给信息加密，在某种程度上，它们只能被相应私钥的持有者阅读，但是在密钥权需要被支配的情形时仍然有一些应用。

12.4 零逼近方法

零逼近校验是不用揭示信息本身就可以知道一些信息的校验技术。这个概念始于20世纪80年代，那时密码学家和理论计算机科学家正在开始探索一种信息能在任意时候被隔离和暴露的方法。从某种意义上说，一个零逼近校验就是一个数字信号的偏激版本。

这里有一个零逼近校验的简单的例子。设G为一组节点$\{v_1, v_2, ..., v_n\}$和连接这些节点的一组棱边$\{(v_i, v_j), ...\}$的曲线图。如果有某种方法把k种颜色分配到n个节点以便使连接两个节点的棱边具有同样的颜色，那么这个曲线图就叫做k色图。那就是说，不存在这样一个i和一个j使当$f(v_i) = f(v_j)$时，(v_i, v_j) 在一组棱边里。F就是色彩函数。图12.3展示了一个四色彩曲线图。

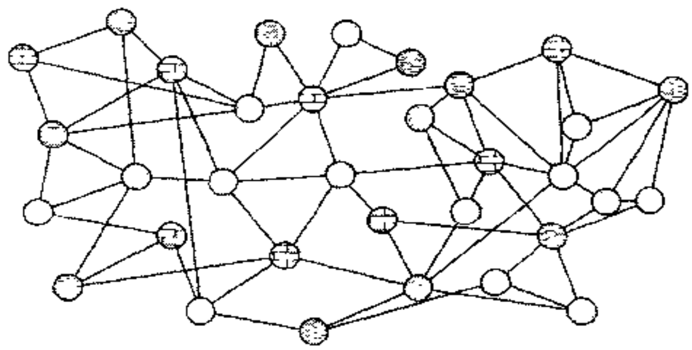


图12.3　所有节点总共有四种颜色的曲线图

在一些情况下，找到任意一个k色彩图可能是一个复杂和困难的过程。这个问题以NP-complete而著称[GJ79]，它意味着例图会随着越来越多的节点和棱边被加入而越来越困难地按指数规律增加。在实践中，很多只有一种色彩的图形可以相对很快地被找到。而找到复杂图形则更困难，这些图是在应用中使用零逼近校验的一种实践限制。

设想你知道某个曲线图的一种颜色并且你想证明你确实没有把它暴露给其他人（怀疑的询问者）。这里有一个简单的证明方法：

1. 产生一个随机的颜色排列f'。也就是说，用一个随机的方法交换不同的颜色以便对所有曲线图中的（v_i, v_j）有$f'(v_i) \neq f'(v_j)$。
2. 怀疑的询问者可以给你一个随机的位串S，这个S可以通过混编一个不易腐蚀的文件而从无懈可击的资源中建立。如果零逼近校验将被嵌入到一个文件里，这个混编就是文件中不被嵌入改变的那部分。
3. 产生n个随机密钥，p_1, ..., p_n，使用一个加密函数给串$S + i + f'(v_i)$的每一个节点加密，图中+代表串联。传送被加密的版本给怀疑的询问者。
4. 怀疑的询问者从图（v_a, v_b）中随机地选择一个棱边并且把它呈现给你。
5. 你把p_a和p_b传送给怀疑的询问者，然后由他使用它们给被加密的颜色版本$f'(v_a)$和f'（v_b）解密。如果这两个译本不同，那么怀疑的询问者就知道你已经知道怎样给至少一部分图上色了。图12.4展示了两个被暴露的毗邻的节点。

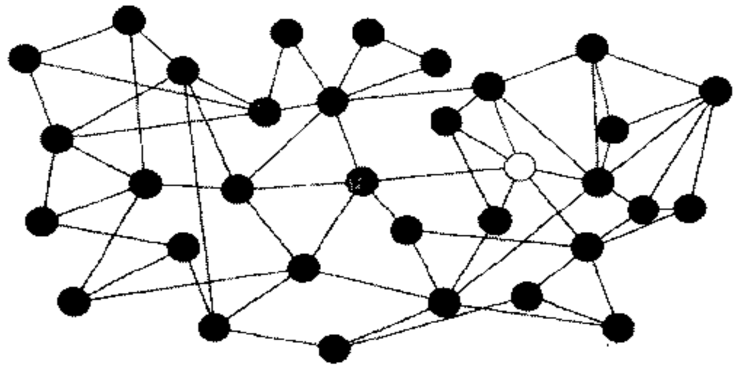


图12.4 在一个零逼近校验中，节点的颜色被加密并且被送到怀疑的询问者那里，他要求的是有两个被暴露的毗邻节点

这个过程应该不断地被重复直到怀疑的询问者满意为止。如果棱边真地是随机选择的，那么在上色中的任何错误都代表每一个步骤被暴露的等价机会。随机性阻止了证明者参与选择f。最后，任何弱点和错误上色都将显示。

像这样的零逼近校验在隐藏信息的世界里是很有用的，因为它们允许信息隐藏者控制怎样及什么时候把信息暴露给另一个人。尽管如此，校验的结构还是保证了信息是不暴露的。这种校验不能被其他的人重复使用作为删除信息的基准。这个解决方法对水印应用也是有用的，版本持有者想证明他们没有把技术暴露给其他人，他们是正当的拥有者。

在理论上，任何零逼近校验都可以被用来作为一个水印。每一个校验都是以一组确认问题实例的位开始的。设想上面所述的加密图形程序的m次重复操作已经足够证明图形颜色。用以下步骤可实现校验在一个文件中嵌入：

1. 通过混编在嵌入过程中不被修改的文件部分来产生一个数值S。这些可以是最小有效位或一些其他的文件高通过滤器。
2. 产生色彩的m排列，f_1'，f_2'，…，f_m'。
3. 产生$m \times n$矩阵密钥，$p_{i,j}$。
4. 通过加密串$S+i+j+f_i'(vj)$的方法给每个校验重复操作中n个节点的色彩加密。这里+代表串联，i代表证明算法（$1 \leqslant i \leqslant m$）的重复，$j$代表节点，（$1 \leqslant j \leqslant n$）。使用$p_{i,j}$。
5. 嵌入每一个被加密的颜色并描述文件里的曲线图。如果需要的话，加密色彩可以被$p_{i,j}$嵌入在隐藏位置。例如，这个数值可以被用来作为一个选择一串像素，并在里面隐藏信号的加密的安全随机数字生成器的根据。

任何对文件有兴趣的人都可以询问信息隐藏者怎样把隐藏的图形里面的颜色根本地去掉。假设这个问题是很难解决的，并且只有那个隐藏图形的人才知道怎样上色，怀疑者可以重复步骤m次。设i代表重复操作。

1. 怀疑者恢复图形。
2. 怀疑者通过检查文件中不改变的部分集合成S。
3. 怀疑者选择一个随机棱边，（v_a，v_b）。

4. 检验器揭示出$p_{i,a}$，$p_{i,b}$。这些值可以用来确定加密色彩的位置并然后解密它们。如果它们不相匹配，过程就继续。如果相匹配，校验就停止。

如果一个合适的曲线图被找到，这个解决方法就可以提供很多优点。隐藏信息的人可以证明他或她隐藏信息时并没有把足够的信息透露给其他的人复制。每一个怀疑者将随机选择棱边，并且所有的棱边都被重复是不可能的。

这个静态模式并没有交互式模式的功能强，因为检验器在上色中必须锁定。如果一个怀疑者可以使检验器多次重复算法，最后，怀疑者将收集足够的密钥，$p_{i,j}$，这是证明她知道图形色彩的一个好机会。最终，检验器将揭示出所有的色彩，这个信息的缓慢泄露意味着校验并不是零逼近的。如果m和n足够大并且校验被执行的次数足够少，那么这个过程仍然是有实践意义的。

找到一个合适的图形并不是一个很容易的过程，有很多种不同的零逼近校验可以在相似的情况下被嵌入，但是它们中大多数都不是特别有实践意义的。校验依赖于最差的条件下的最难的问题，但是通常是相当简单的。找到并确认难题实例并不容易，如12.3.1所注，一个早期公钥系统是由Ralph Merkle和Martin Hellman使用以渐缩算法而著称的NP-complete的问题而产生的（给出n个项{w_1，...w_n}的加权值，找出一个加权值加起来是一个特殊值的子集）。最终，所有的算法模式都被破坏了，并且很多已成长为基本的NP-complete问题，据信能用来产生难题实例。

12.4.1 校验的离散对数

Scott Craver发展出一种依赖于离散对数问题难度的零逼近水印法机制（给出y，找到一个x使得$y = g_x \bmod q$，这里q是一个质数）。这种方法的成功在于运用了大量的伪造水印作为假目标。理想上，任何试图篡改水印的人都必须像移除所有的假目标那样移除一个真正的水印。最佳的效果就是有如此多的信息以致于移除它们实质上就等于改变整个图像。

令x为一个水印密钥，这个算法将允许信息隐藏者证明他或她是在未泄露x本身的情况下知道x的。检验器通过使用一个密钥p_1选择文件里的特定区域，从而在文件中隐藏$y = g^x \bmod q$。

然后检验器选取偶然存在于文件中的$n-1$伪造水印，即检验器选择$n-1$个密钥，P_2，...P_n，来抽取$n-1$个数值，y_2，...，y_n。使用来自于文件中的伪造假目标可以帮助增加水印的强度。有越多的假目标存在，移除它们就越可能损坏图像。在某些情况下，如果这使得任务更容易完成，那么检验器就可更实际地嵌入它们。

当一个怀疑者展示文件的时候，检验器不用给怀疑者重复过程的权利就可以显示出x的真值。假目标阻碍了怀疑者认识y_i的值有一个已知的对数。这里步骤如下：

1. 校准仪给怀疑者密钥，p_1，...，p_n。
2. 怀疑者恢复y_i的值。
3. 如需要，这个循环可被重复多次。
   A. 检验器产生n个使人糊涂的因素，b_1，...，b_n，它们被用来给水印加密，方法是计算$w_i = g^{b_i} y_i$。
   B. 检验器在把数值w_i传送到怀疑者那里之前先加密这些数。
   C. 怀疑者交换一个命令信息并使两个命令中的一个运行在检验器上：

a. 告诉我所有n个使人糊涂的因素，b_1，…，b_n的值。知道了这些，怀疑者就能检查并确信对每一个值y_i都有一个相对应的合理值w_i。

b. 检验你知道这些值的对数，检验器通过揭示$x+b_1 \bmod (q-1)$来完成校验，在这是一个$w_1 = g^{b_1}y_1 = g^{b_1}g^{x} = g^{x+b_1} \bmod q$的对数。

在每一个重复操作里，检验器必须揭示一半的方法。这种结构使得怀疑者没有回转的余地，并且不能把校验成功地重复给其他人。从一个操作中得到答案并使用信息反馈给其他人是不可能的，只有知道x值的人才能做到。

这种算法仍然有其局限性。检验器必须揭露不同位块的位置，这使得它们被怀疑与损坏有关。假目标既增加了攻击者的工作量，也增加了移除所有信息实质上就等于改变了文件的可能性。

提高这种算法力度的一种方法就是把许多真正的目标和假目标混合在一起。即检验器安排隐藏复合值y_i，这里我们已知x_i的值使得$y_i = g^{x_i} \bmod q$。只要检验器知道数值子集中的x_i值，它就可以知道整个数值子集的值。当然，复合拷贝和复合块能帮助减少危险，但却不能消除危险。最终，所有的目标都将被揭露。

12.5 串通控制

因为缺乏一个更好的术语，所以“加密钥”水印偶尔被称做“串通控制”。假设你建立了带有两个水印位w_1和w_2的两个文件d_1和d_2，w_1和w_2的作用是给真正所有者的身份编码。水印学是一门不完善的科学，所以尽管你尽了最大的努力，某些人还是试图欺骗系统并对比文件发现它们的不同点。欺骗者可以引入随机噪声到这些位置，有效地改变一半的位。这会给水印带来什么样的后果呢？

基本系统将会毁于一旦。如果w_1 = 001010111和w_2 = 001010100，它们只有最后两个位数是不同的。一个新的水印，00101010101，就将会产生并牵连到其他一些人。

串通控制系统可以解决这个问题，串通控制系统首先是由Dan Boneh和James Shaw引入的[BS95]。他们的这个机制就像是误码校正代码的延伸。

误码校正代码在第3章里描述。

这里有个简单的例子。设S是只有一个单独位1的n位代码字组。如果n = 4，那么S = {1000，0100，0010，0001}。设每一个文件被分配一个代码字，如果两个文件的所有者串通，他们就只能找到不同于两个位的水印。Boneh和Shaw称这为一个“结构校验”代码，并注明任何n个用户的“结构校验”代码都都必须以n位为开头。所以这种构造是最理想的。

例如，设Alice的水印标志为0100，Bob的水印标志为0001，有人企图通过建立一个把两个文件混合在一起的合成文件来擦除水印，在确认了所有的位都不同后，攻击者选择Alice文件的一半和Bob文件的一半。如果有人想通过比较Alice文件和Bob的文件来找到不同之处，那么会发生什么情况呢？如果有人用水印标志0101建立一个新文件，那么Alice和Bob都会被牵连其中。任何检查这个文件的人都能追溯回这原本的两个文件。改变50%的位会产生两个水印标志中的一个，另一个将保持受牵连的状态。

当然，一个聪明的攻击者可能会把两个文件都置换到0以产生一个水印标志0000，这不会使任何一个受到牵连，但水印标志0000是不容易产生的。任何人都相信水印标志不会选择像这个例子一样的简单的位向量。他们可以用一个密码XOR（异或）它们，攻击者不知道什么是向上、向下。所以说，第4位置的一个零可以在Bob文件被密码向量异或后确认出它。

Boneh和Shaw把这个思想和误码校正代码世界的思想结合起来并进行了延伸，因为误码校正代码会弥补随机交换，所以任何一个攻击者在水印标志被擦除之前，都必须置换很多位。

12.6 小结

本书里的大部分算法都用一个密钥添加了额外的安全性。通过计算一些加密或混编函数：*f*(*key*)，*f*(*f*(*key*))，*f*(*f*(*f*(*key*)))，...的连续数值，这个密钥就能被用来产生一个伪随机位流，这个位流既可用来选择文件的一个子集，又可用来控制它怎样在各个位置被加密。只有有同样密钥的人才能恢复信息。当然，在文件被嵌入之前用另外一个密钥给它加密也是有意义的。

加密钥算法抵抗了计算机标准的普通公共特性。如果这个软件经常被使用并分配给许多人，那么它里面的算法会变得大众化。给这个算法加个密钥就可大大加强它。

- **伪装**　许多算法通过使用密钥修改算法的行为来添加一个额外的伪装层。这就意味任何攻击者仅仅用算法的知识是不可能选取数据的。
- **安全性**　通过重复的加密或混编创建的伪随机位流是相当安全的。
- **怎样使用**　用一个伪随机密钥流量来代替用来添加噪声的任意一个随机数字生成器。这个被加密钥的随机数字流也可用来选取特殊的子集或重新安排数据本身。例如第13章的算法，使用一个被加上了密钥的加密函数来改变数据的顺序。

第13章　排序和重排序

为什么要进入前10名？以下列出的是失败的头10个原因：

10 谁想被列在表的后面呢？再说能在表中有一席之地总比排在第11位要好。

9 其实排在第9位真地要比第10位好吗？

8 常有20～30个人说他们“在前10名”中。

7 “列表提供了一个顺序的假象，这个假象具体化了我们关于无序这种地狱般的势力的根本焦虑。”这是Montparnasse博士的理论。

6 第6位是一个比较舒适的位置，它不用因排位太高或太低被忽略过而担心，列表中的位置跌落是很糟糕的。

5 五个金环。

4 没有数字4。

3 所有好事都来自于前3位，但是如果你只是10个中的第3个那就不一样了。在100名中排第3名是非常好的，当然，每个前10名都是因为除去了100中的其余90个或1000中的其余990个，然而，这界线你要划在哪儿呢？

2 没人会记得银奖获得者。

1 如果没有一人要枪杀你从而取代你的位置，那么位于第一名是比较有意思的。

13.1 说明

本书中的大部分算法都用相对严格的数据格式将信息隐藏起来了，图像文件必须用一个定义完整的次序描绘每一个像素的颜色，音频文件对每个声音节点的描述有很高的要求。因为文件将不重新排序，所以将信息隐藏在噪声的特定位置上是一个相当好的投机办法。

有些数据不是如此严格的，文件可以不需要变换整体意思而重新排列章、节、段甚至句子的顺序，这就是说，一个文本文件的整体意思不会因为句、段、节或是章的重新排序而改变，这就可以不被读者发觉，本书中的不少章就是在不影响全文的流畅性的前提下进行的一些新的排序。

甚至那些有很强空间、时间联系的文件也可以重新排序。图像文件可以被剪裁、翻转或者安排在一个区域中，歌曲甚至可以在被混合在一个文件里后还能被重新排列和编辑。所有这些改变都可以重新排序文件。

当数据能被被重新排序时，攻击者就可以不扰乱表面数据而破坏隐藏的信息，**Mikhail Atallah**和**Victor Raskin**在设计一个在自然语言文本里隐藏信息的机制时，就面临这个问题。如果他们通过改变每个句子来隐藏一个或两个位的信息，那么他们在每个句子的顺序被重新排列[ABC+01]时就会失去全部信息。第6章、第7章和第8章的很多文本机制对于这种攻击是很脆弱的。数据流量起始处的一个变换能混淆它之后的每一个解码。在某些伪随机通量数值

的指导下，第9章中的很多解决方法可在随机的位置储存位。流量中较早毁坏的位会使跟随其后的其余文件变成一堆垃圾。

信息流由两部分组成：数据和存储数据的位置。本书中的大部分算法都集中在设计和遮盖数据直到数据采用正确的伪装。数据所在的位置很少有多于一个的增加安全的工具，并且这些算法很少关注这个问题。

本章的焦点是第二个组成成分：如何保护和利用信息存放的位置，本章中的一些算法为了希望隐藏在底层的信息离开原位而将数据隔离，以此来抵抗那些移除文件的攻击者。他们提供了为一个文件建立规范顺序的一个简单的方法，尽管攻击者们可以随心所欲地将文件重新排序，但收发双方仍能重新建立那个规范顺序并发送一条信息。

其他算法利用内容本身的顺序而不改变潜在的信息。n个信息被安排可以有$n!$种方法。这就意味着在有n项的一个清单中有$\log_2 n!$位能被传送。这些信息中没有一个会自己改变它们在清单中的顺序。

仍有一些其他算法混合在假的数据中作为诱饵，他们按照自己的方法运算直到他们被清除和规范顺序被恢复。

全部的算法都依赖这个事实，即如果正确的顺序能在后来被找到，信息就不需要在一个预置的模式里流动。如果信息的所有部分都是沿着不同的路径传输，那么这个事实也是很有用的。

13.2 编码强度

隐写术系统中的许多攻击试图用微小的重新排序来破坏信息。图像扭曲是最复杂的和最令人气馁的搅乱基准的一个方法，这个基准被用来在图像文件中衡量隐藏数据的技术是否足够强大。

这里是一个非常抽象的方法概要，这个方法使任意一个隐写术系统能抵制重新排序。

1. 将数据流分成离散元素：$\{x_1, x_2, \ldots, x_n\}$。它可能是文字、像素、像素块或文件的任意子集。这些块都应足够小使其能经受敌人的任何扭曲和搅乱，但又要大到可以用于区分。
2. 选一个函数f，它是一个独立的变量，可将它用在文件隐藏过程中。如果最小有效位被改变用于隐藏一个信息，那么f应该仅依赖于其他的位。

   很多用于设计哈希函数的相同原则也可以用来设计这个分类函数。这个函数f也可以键入，这样一个额外的数值k能够改变结果。

在一次性的哈希系统中，发送方更换用于掩饰的数据x，直到$f(x)$发出正确的信息[Shi99]。因为寻找正确的x要求强制力量，所以这个信息通常和1位一样短。这与第9章中变换每个像素的颜色直到类似地发送正确的信息的机制很相似。

3. 将f运用到各元素中。
4. 按f将清单进行分类。
5. 用合适的方法将信息隐藏在每一个元素x_i中。

6. 为使接收者通过任何方法都能获得全部元素$\{x_1, ..., x_n\}$的数值，不要对表进行分类或用某种方法整理。

信息可以隐藏在复合的位置并进行重排序，而不需要与信息的源端进行交流。惟一需要的同步就是f的选择和可能的密钥k。有一个好消息，就是复合的内容能按照不受约束的进程，通过不同的通道，以任意的顺序到达，并且不会损坏信息。

分类的数据允许发送者以任意的方式搅乱表面的数据，当信息到达时，接收者能正确地重新排序它从而保证信息的安全。

分类也是一条很好的可取之道，它可为保留隐藏信息选择一个表面数据子集，别的一些解决方法是使用某些密钥和一个随机数字发生器来选择一个随机的信息子集。还有另一个解决方法是在数据流中运用某些元素的f函数、分类数据流，然后选择第一个n来保留隐藏的信息。

当然，这些方法的成功完全取决于函数f的选取。一旦数据被转换成某些数据格式，很多问题就可以被解决。它已经是一个很大的数，所以分类数值是很容易的。这个函数$f(x)=x$本身通常是足够好的。

有一个潜在的问题会发生：如果多个变量是完全相同的，或者产生相同的f值，也就是说如果存在i和j，假如$f(x_i)=f(x_j)$。在许多普通的分类操作中，i和j被用于破坏纽带，但这做不到，因为收发双方可能会以不同的顺序获得x的值。

如果$x_i=x_j$，那么它们出现的顺序是不碍事的。但如果x_i不等于x_j，则问题就会出现在收发双方是否使用同一顺序放置它们。在接收方从x_i和x_j中获取数据后，它将按一个错误的顺序结束，潜在地搅乱了获取的其余信息。

有两种解决这个问题的方法。最简单的一种当然是只有当$x_i=x_j$时，$f(x_i)=f(x_j)$。只有当f是一个加密结构函数时，这种情况才会发生。这通常是最好的解决方法。另一种方法是用某些错误校正代码来移除由解码问题造成的损坏。某些错误校正代码可以很好地处理一行中的几个错误，这是非常必要的。如果因为$f(x_i)=f(x_j)$使x_i和x_j失序，那么它们会产生两个错误。如果有复合数值产生相同的f，则就会存在复合的问题了。

如果不可能保证f的数值是惟一的，那么保证相同数量的信息被打包分组在每一个元素x_i中是很有意义的，这就保证了由错误排序引起的损坏不会扰乱其他的数据包。

13.3 恒定形式

在创建常量函数f时，一个最大的挑战就是：信息编码器将在数据被隐藏之前使用f选择顺序，而接收者则是在数据被插入之后使用f。如果隐藏过程改变了f的值，那么顺序将被破坏。

这里有两种基本的设计f的方法，首先保证数据隐藏时f不会发生变化。一些简单的常量函数如下：

- 如果元素是数据被插入的最小有效位的像素，那么f的值应该包括计算结果中的最小有效位。
- 如果元素被压缩成音频或图像数据的版本，那么函数将包括可能被插入信息而改变的系数。例如，JPEG文件可以通过修改系数的最小有效位来隐藏数据。函数f应该依

赖于其他的位。

- 如果数据被隐藏在一些具有一个可以在不超过±ε范围内修改个别元素的宽频技术的元素中，那么数值可被标准化。让x_i成为一个元素的数值。这定义了一个范围：$x_i-\varepsilon<x_i\leqslant x_i+\varepsilon$。设一个标准的或者规范的点为{0，2ε，4ε，6ε，...}，这些点中有且只有一个能被保证在各自范围内。每个x_i的值可被一个位于±ε范围内的点来取代。

为计算f，使用规范点代替真正的x_i的值。

13.4 标准形式

另一种解决方法是为每一个元素创建一个"规范的形式"。这就是说，选择一个总是相同的元素译本。然后，通过把它传送进规范形式来移动数据；其结果就被用于计算f。

这里有一些规范形式的基本例子。

- 如果数据被隐藏在像素或音频文件元素的最小有效位中，那么通过把最小有效位设为0而找到规范形式。
- 如果信息隐藏过程修改不多于±ε范围内的元素，那么规范点就可以和上面一样被建立。使{0，2ε，4ε，6ε，...}成为一组规范点。只有一个将位于$x_i-\varepsilon<x_i\leqslant x_i+\varepsilon$的范围内。
- 句子能被写成规范的形式，Mikhail Atallah和Victor Raskin使用自然语言处理算法来分析句子[ABC+01]，句子"狗追猫"在它们的基于lisp的系统中采用如下形式：

```
(S
(NP the dog)
(VP chased)
(NP the cat)))
```

字母"S"代表一个句子的开始，字母"NP"代表名词短语，字母"VP"代表动词短语。如果这句子不能被分析，它就被忽略。

他们的解决方法是通过应用大量的比如在主动和被动语态之间的转换来隐藏信息。通过把这个例子换成"猫被狗追逐"的被动语态，一位就被编码。他们使用的其他方法包括移动一个句子的节点，分开句子，和插入像"这看上去是……"的额外不需要的字和词 。

一个规范形式通过选择一个转变译本而被定义。在这个例子中，主动语态可能是句子的规范的形式。

13.5 复合信息的分组

分类也可以让数据集中储存复合消息。如果f_1定义一个顺序，f_2定义另一个顺序，那么如果大小尺寸正确的话，两种顺序都可被用于隐藏信息。如果伪装数据的元素数量比已经隐藏的信息数量大，那么发生冲突的机会是很小的。

如果采用分类函数f_1和f_2，它们像随机数字生成器一样工作并且用相等的概率按顺序放置每一个元素，那么一个冲突的发生的机会是很容易估计的，最简单方法就是使用一个设计

完备的、密码安全的哈希函数，如SHA。设计者已经尽力保证这些函数结果接近于随机数字源。

如果采用的f_1和f_2足够随机，则一个冲突的奇校验是简单的。如果数据被隐藏在由f_1产生的清单里的前n_1元素和由f_2产生的清单里的前n_2元素中，那么被估计的冲突发生机率是：n_1乘以n_2除以n的商：

$$n_1 \times \frac{n_2}{n}$$

通过使用像第3章描述的误码校正编码可减少损坏。

13.6 隐藏信息的分类

在本章中第一部分的算法使用分类作为一个伪装形式，信息被隐藏并且为了多隐藏些而被搅乱。正确的信息分类可再次揭示它。

另一种解决的方法实际上是根据加密的选择来隐藏信息。假设这里有n项，则有$n!$种方式来安排。如果这些安排已被给定了数字，那么就有$\log n!$位。Matthew kwan使用这个方法通过GIF（可交换的图像文件格式）图像的调色板里的颜色来隐藏信息。一个可免费获得的、名为Gifshuffle的程序能实现这个解决方法。

这里有一个简单的算法译本：

1. 使用一个被加密的伪随机位流量选择像素对或文件的数据消息。
2. 对每一对数字来说，D表示它们之间的不同。
3. 如果这些不同比感知临界值还要大，则可忽略这一对。这就是说，如果像素之间或数据项之间的不同会引起一个人的注意，那么就忽略这一对。
4. 如果$D=0$，则也忽略此对。
5. 逐个配对，为信息的每一个位编码，使$D>0$代表0，$D<0$代表1。如果像素不能按正确的顺序安排，则交换它们。

这个基本的机制将位隐藏在成对像素或数据消息中，这种方法不会改变文件原先的基本统计轮廓，因为许多对隐写术系统的攻击都依赖于统计分析，所以这是一个重要的考虑因素。当然，它会改变某些大的有关某些数值的像素与其他不同数值的像素相邻近的统计。寻找基本统计的攻击者不会查觉出变化，但更熟练的攻击者可查觉到。

> 电路中晶体管的位置能按不同方法分类，以此来隐藏信息，这样可能以追踪到公正的所有者[LMSP98]。

这一算法以可以被修改以隐藏信息统计值。一个名为Patchwork的早期隐写术算法多次地重复了这个过程以便在大量配对中隐藏相同的位。这个随机过程通过从一组中选一个像素，又从另一不同组中选择另一个像素来选择配对，信息通过比较两组中的统计的不同而被察觉，最大的一个识别的位被传送。这不存在使像素随意选择达到同步的企图。

在这个简单的例子中，一个位按21的顺序被隐藏起来。这个过程可通过选择n个像素的组成或项以及他们全部的排序来隐藏信息。这里有一个方法如下：一个n条消息$\{x_0, x_1, \ldots x_{n-1}\}$的组，可编码成一个长的$\log n!$位数值$M$。设$m=M$并让$S$为一组$\{x_1, x_2, \ldots x_n\}$，设答案为A，

几初始为空组。重复这个环路i，使数值从n开始并减少到2。

1. 在S中用索引m mod i选择项。指数从0开始直到$i-1$，每次经过循环，s中只有i个元素留下。
2. 从S中去掉它，在A的结尾保留它。
3. 设$m=\frac{m}{i}$，循环至最接近的整数值。

通过这个环，m的值可被恢复，从$m=1$开始。

1. 去掉A内第一个元素。
2. 计算A的左边的数值（有一个比A本身小的下标）数值，可以将这个元素转变成一个数值。就是说，如果你去掉x_i，则计算仍然留在A中j小于i的数值x_j。
3. 将这个数与m相乘，并让它成为m新的值。

使用这个算法通常要求找到一个给元素分配正确顺序的方法。这个算法假定S组中的元素按序从0取到$n-1$。例如，kwan，使用叫做“自然（natural）”的顺序将RGB（红、绿、蓝彩色值）颜色在GIF调色板中排序。就是说，每一种颜色被分配数值为$2^{16}\times$红色$+2^8\times$蓝色＋绿色，以及按此分类。

很多使用以上方法的分类策略也可以被用于元素分类，如果在分类前用一个特殊的密钥将它们加密，则这钥匙就作为这个隐写术的密码。

这个算法在合适的情况下是很有效的。想像在清单中你有65 536字节数值要安排，这占了128KB。用此算法——接近90%的负载，可使清单存储119 225字节的数值。当然，像购物单那样的好的机会是很少有如此紧凑的，但是理论上的潜力仍很惊人。

13.7 添加额外数据包

另一种打乱顺序的方法是添加伪造数据包。Ron Rivest提出这个想法作为一个在隐写术中[Riv]逃避禁令的方法。他建议让每个包与数字信号相联系并不受阻碍地流通，有效的包带来有效的信号，无效的包带来无效的信号。

Rivest建议使用由哈希函数产生的被加密的信息确认代码。如果x是信息，则只有读信息的人知道信息是由密钥为k的$f(kx)$组成的。让$f'(x)$成为某些产生坏信息的函数，可能是一个随机数字发生器，使信息成为$\{x_1, x_2, ...x_n\}$。Rivest称这些信息为“小麦”，使“谷壳”为$\{r_1...r_k\}$，正确大小的任意数量。信息由像掺有如$(r_i, f'(r_i))$这样的分散注意力的东西的$(x_1, f(kx_1))$而配对组成。

如果我们放松信号能被任何一个可成功访问一个公共密钥的人所测试的观念，则很多标准数字信号算法也可以被使用。如果确认钥匙是保密的，则只有发送方和任意接收者能够从谷壳中辨别出麦粒。因为接收的密钥不能被用于产生它们自己的信号，所以传统的公共密钥在这里还是有用的。

> Mihir Bellare和Alexandra Boldyreva用更高效的、完全效果的变换进一步发展了谷壳想法。

当Rivest的文章[Riv]被写出来时，美国政府限制了以“密码术”命名的算法的出口而却没有限制“确认”算法的使用。基于哈希函数的信息确认代码，被典型地假设为是不提供保

密措施的，于是这样它们就可以自由地出口。Rivest提醒他的方法指出了一个使法律效力变弱的漏洞。

这种解决方法的安全性更大程度上依赖于x的结构及潜在的数据。如果x包括了令人感兴趣的足够信息，那么袭击者就不必担心f或者k就可以从谷壳中获的麦粒。

有一个解决办法就是将文件分成独自的位。不管怎样，因为这只有两个有效信号：$f(k0)$和$f(k1)$，这个机制是有点不可靠的。Rivest通过添加一个计数器或现时的混合，以使每个包看起来像（x_i, i, $f(kix_i)$），从而克服了这个问题。因为每一个位都有可能要求一个80～200位长的数据包装载它，所以这个机制不是很高效的。

这个解决方法容易与其他定义顺序的技术相混合。一个f函数可以识别真正的信息元素，另一个函数g可以识别元素规范的顺序以便能取得信息。

13.8 小结

一组目标的顺序是一组令人吃惊的复杂信息，它为隐写术提供了一个相应的巨大机会。有时某个敌人为了寄希望于破坏某些隐写术而会选择变换对象清单的顺序，有时候数据体选取不同的异步路径。在任一种情况下，被某个加密函数f定义的隐藏规范顺序是一个存储顺序和抵抗袭击者的一个好办法。

这个想法也可以为储存信息开个头。除了很多在17章中描述的隐写术的统计或结构形式以外，改变目标顺序要求对象本身不发生改变。微妙的变化不会产生异常。这结果可保留令人惊讶的大量的信息，n个目标能存储logn!位。

当然，有时微小的变化要被用于添加覆盖物。很多列表被按字母顺序或某些别的数据域分类，这个域将会是一个人为产生的、作为覆盖物的系统。例如，为了掩饰分类的原因，一个人数的清单可以通过随机产生的成员数字来分类。

- **伪装** 任意n项的组$\{x_1, ..., x_n\}$可以按分类的顺序隐藏logn!位的信息，通过当f是一个加密术或一个哈希函数的地方，用$f(x_i)$代替x_i分类的方法可以使这些信息被上锁或隐藏。

  如果另一个函数g能用于产生一个安全的信号，或区分有效包和无效包的信息确认代码，那么作为诱惑的包就可分散偷听者的注意力。

  有时候袭击者为破坏一个信息，会重新安排目标组的顺序。因此，在处理它们前将目标用函数f分类是一个好的防御方法。
- **安全性** 安全性依赖于伪装数据的质量和函数f的安全性。如果目标组看起来好像是一个完全随机的清单，那么它就不可能接受任何仔细检查。如果攻击者发现这一个组的真正的顺序——一个固定加密的或哈希函数可有效防止的一些东西，这信息就只能被榨出。
- **怎样使用** 为了它们的清白，选择你的目标，选择一个代码函数，选择一个密钥，为全部的目标计算$f(x_i)$，为得到所谓的真正的顺序而分类这个数值，使用重排序函数插入数据，并且用这个顺序运送它。

第14章 传 播

一项新的工作

我们以拍摄到的一只渔船展现给大家，在渔船上一个父亲和他的儿子正在工作着，他们正在用蚯蚓和小鱼引诱鱼上钩。

父亲： 我的儿子从大学回到家乡并且和我们呆在一起，这让我非常开心。

儿子： 哎呀！该死。

父亲： 不。说话礼貌点。我想看到我花的钱所得到的是什么（指供儿子上大学）。

儿子： 该死。

父亲： 不。你过来。

儿子： 一个文字的本质语言是混乱、分散、误用和伪装。只有通过解析原文的表示者，估计和再估计记号路标，然后解析连贯的分歧的视觉，我们才可能开始真正地领悟和我们文学媒体的限制。

儿子绕着他的鱼杆线并且开始换诱饵。

父亲： 我很自豪能够听到这样的话。

儿子： 我只是在重复而已。

父亲： 英语文学学位从某种意义上说也是令人自豪的。

儿子： 如果你这样说，那就算是。

父亲： 现在，你准备去找什么样的工作？

儿子： 哦，没有很多工作是对主修文学的毕业生开放的。我有一个朋友是在纽约的一个出版社工作……

儿子的声音变小了，父亲停了下来，在座位上踌躇了一下，调整了一下他的鱼线并且开始钓鱼。

父亲： 做那些事是没有钱的，现在的大学不是过去的大学。我和你的舅舅说过，他也同意我的看法。我们都想你加入我们家的生意。

儿子： 是进口还是出口生意？这又和文学有什么关系呢？

父亲： 我们不是技术上的进口和出口。

儿子： 但是。

父亲： 是的，就像商业的名词所说的，但是当你加入的时候，文字里会有一点点混乱、分散和伪装。

儿子： 但是成箱的番茄和无止境的塑料商品流通又怎么说呢？

父亲： 我们是真正地在移动着金钱，我们是专门研究洗钱的。

儿子： 金钱和番茄之间又有什么关系呢？

父亲： 既可以说没有关系也可以说密切相关。我们运输番茄，他们为此支付我们金钱，某些人卖番茄，并且最后每个人都很高兴。但是当每一件事被加起来时，当做

完所有的算术时，我们已经为我们的朋友转移了超过100万的美元。他们看上去像是使他们的番茄获得巨大利润的番茄国王。我们得到的是我们自己的那部分。

儿子： 所以这一切都是一个伪装？

父亲： 如果你认为番茄是一个装满错误方向、错误领悟和错误理解的语言的话，它应该是更高级的。我们在金钱语言方面做出了一个很清楚的陈述，把它转化成数百万个微小的番茄句子，并且然后通过魔术般的计算把它转变回冷语言，硬币。

儿子： 这是一种新的语言。

父亲： 这是旧的语言。你要理解加法，你要理解总和大于部分，那就是我们需要做的。

儿子： 那你想从我这里得到什么呢？

父亲： 你现在是一个指导错误的语言专家，那是我们的办公室需要的。

儿子： 一份工作？

父亲： 是的。我们可以支付你3倍于出版社提供给你的薪水。

儿子： 但是那样的话，我将不得不忘记我在大学里所学的所有字。我还需要会适当的搬运工和港口工人的语言。

父亲： 不。我们只想你继续这样保持下去。所有的文学语言都要比我们建立的任何代码更令人混淆。

父亲和儿子高兴地拥抱在了一起，结局没有被混乱或指导错误所损坏。

14.1 信息传播

本书的很多算法都在位是位，0是0，和1是1的现代数字时期中进化发展的，所有的工具和解决方法都假设信息将作为二进制数流量被编码。因为现代机器能做的最好的就是二进制数字，所以调整这个应用到狭窄的世界观是很重要的。它们仍然具有吹毛求疵的精密度的能力，但是它们却是通过求大量位数的近似值来完成它的。

本章的算法采用了一个稍微不同的方法。当信息仍然被编码为0和1时，理论上还包括更多的连续的制式。每一个位置的信息都有很小数量的变化，它们可能是像0.042或1.993那样的小数。这个细节最终通过四舍五入而消除了，但是理论上还是包含这些小数值。

这在很大程度上是因为这些算法模仿了一个由无线电工程师建立的完整技术集合。过去，当无线电工程师大量地使用模拟技术发展频谱传播的无线电通信时，他们就已经攻击过一个隐藏信息的相似问题。在一开始，无线电广播设备是通过以一组频率把能量"抽进"或"抽出"（译者：即电磁转换）它们的天线的方法来实现传播的。通过稍微地改变能量，信号就得到编码，调幅（AM）无线电广播改变信号的强度，而调频（FM）无线电广播则是调制信号的频率。用无线电工程师的行话来说，就是所有的"能量"都集中在一个频率上。

调频传播的无线电广播自己转变了这个想法，于是它使用几个频率来代替使用一个频率。所有的能量在一个很大的频率段里被分配——即一个使无线电广播通信更秘密、更可靠、更有效和更少干扰的技术。如果信号是依靠很多不同的频率而存活，那么有意的或无意的干扰破坏信号的可能性就会减小。几个无线电广播也可以不受干扰地共享同样的频率组。也许它们可能会受到一些轻微的干扰，但是这不足以扰乱通信。

调频传播的无线电广播世界里的很多技术都和今天的数字加密学紧密相关，如果无线

电广播术语被翻译成数字语言，那么这些想法或技术就能很好地应用。这并不像它看上去那么复杂。另外，使用数字化能够提供一些额外的益处，这可能就是为什么今天的调频无线电广播是完全数字化的道理。

在无线电广播专门术语里的基本想法，是“通过频谱传播能量”——也就是说把信号放在大量不同的频率上。在有些情况下，系统中频率到频率的无线电电波跳跃是很快的，我们称之为time sequence（时间顺序）。这种频率跳跃与使用一个随机数字生成器来选择一个伪装文件中的位应该被隐藏的位置的技术非常相似。还有些情况是，为了帮助频谱传播的无线电的频率到频率的跳跃而发展起来的随机数字生成器会被用来在一个音频文件里选择像素或瞬间动量。

有时候，这些系统会在同一时间使用不同的频率，即一个被命名为direct sequence（直接顺序）的方法。通过在某一个频率上传播一些数量的信息，信息就被传播到整个频率，对于使用另外一个频率也是如此，等等。通过结合所有的信息，整个信息就被重新集合起来。

能量被分配的方法通常是相当基础的。在每一个例子里，以所有频率被传播的能量加起来产生了整个信号。在数学上这通常用一个整数来代表。图14.1显示了两个假定的分配。上边的图被整化成一个正数，比方说100.03，底下的图被整化为80.2。两个函数看上去非常地相似，它们沿着x轴有同样数量的折曲拐点和相同的零值。尽管如此，上面的模型在x轴上方还是有更多一点的“能量”。当所有的能量都被加起来时，上面那个函数就会发送一个信息“100.03”。

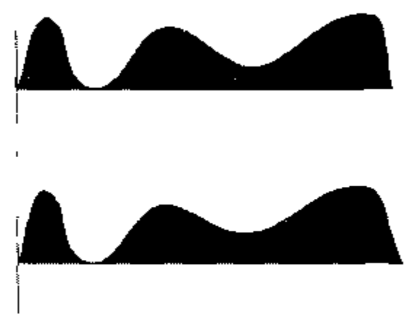


图14.1　通过大量不同频率传播的能量以整数化被计算出。上边的图被整化成，比方说，100.03，底下的图被整化为80.2。它们看上去都非常相似，但是上面的图显得更不稳定

像这样的频谱传播的无线电信号据说可以抵制由无线电干扰发射台引起的噪声和其他干扰。沿着不同频率的随机噪声可以使信号扭曲，但是这些扭曲改变是有可能被删除的。无线电工程师拥有这类破坏无线电频谱的躁声的复杂模型，并且他们使用这些模型来调谐频谱传播算法。噪声可能会在一个频率上增加信号，但是在其他一些地方它却有可能减小信号。当每一个部分都在整数里加到一起时，同样的结果就出来了。图14.2显示了在1位的噪声破坏图14.1里的信号之后所表现出来的信号。

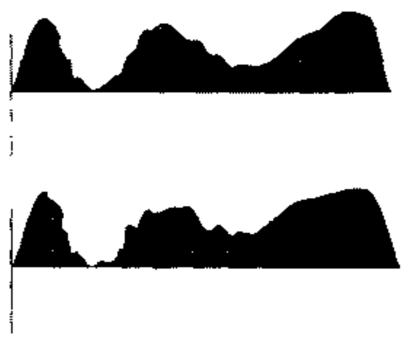


图14.2 图14.1中的信号被1位的随机噪声改变后如图所示。整数结果仍然是100.03和0

一个试图破坏信号的无线电干扰发射台面临着一个困难的挑战。抽空在一个频率上的随机噪声可能会干扰一个集中在一个单独频率的信号，但是它只是模糊了传播于大量频率中的信号的一小部分。通过对每一个频率上所察觉到的能量数进行限制，一个强大信号的影响就可以过滤掉。

如果一个干扰发射台能实际地确认出用于整个调频传播信号的频率带宽，那么通过在每一处插入噪声后，信号仍然还是很难被干扰。噪声恰好被删除了，并且信号通过整数化后一直是卓越的。在实践中，干扰频谱传播信号有多种方法，但是这些方法通常需要干扰发射台抽空更多的、足够的能量，大量的噪声可以破坏系统。

把信息在大量位中展开的算法和第3章描述的误码校正代码很相似。

14.2 数字化

频谱传播算法使用连续函数的模拟比喻来测量被整数化估计的“能量”。在数字世界里，测量任何数的整数定义的离散函数，很多情况下这些整数都可以测量能量。定义任何特殊像素中红、蓝、绿颜色数量的整数，也可测量来自于那个位置的红光、蓝光和绿光。一个声音文件里的整数测量作为压力波传播的能量数。当然，同样的技术也可以应用于测量不同于能量的其他的一般数字。为什么这些技术不能用于一个计算钱的清算账目程序，这是没有原因可解释的。

一个频谱传播数字系统使用以下的步骤：

1. **选择位置**。如果信息在文件里将被展开在很多不同的位置，那么应当尽可能谨慎地选择这些实际的位置。最简单的解决方法是选择一个像素块、一个声音文件部分或者也许是一个数字块。在一些情况下更复杂的解决方法也应该弄清楚。没有原因可解释为什么位置不能被重新安排、重新排序或被一些加密算法直接地选择。用一个分类算法（参看13章）可以找到正确的位置，或者用一个在像素和像素之间跳跃的简单的随机数字生成器也可以找到正确的位置。
2. **确定信号强度**。调频传播解决方法试图在噪声里隐藏信号，如果隐藏的信号比噪声大很多，以致于它能够竞争并压倒主要信号，那么这个方法就失去了它的意义。

选择隐藏信息的强度是一门艺术。一个强大的信号能够抵制不精确的压缩算法的影响，但是这也是值得注意的，一个较弱的信号可以避免被一个临时用户和任何试图恢复它的人所察觉。

3. **研究人的反应**。很多声音或视觉文件的频谱传播方法的发明者曾经研究人的感知限制，并且试图使他们的信号能够被安排在这些限制以外。插进一个音频文件的信号可能是被放在噪声的低电平或者隐藏在回波里。

   当然，人的感知力范围是很宽的。可能还有一些更敏感的眼睛和耳朵，每一个人都可以通过训练得益。很多频谱传播信号设计者发明了能欺骗他们自己和他们的朋友的工具，但是这些工具仍很容易被其他人所发觉。

4. **插入信号**。在所有的位置做一些瞬间的变化，信息就可被加到覆盖数据上。如果文件是一张图，那么声音的每一个瞬间可以得到更强的或更弱的1位。

采取同样的步骤找到文件里正确的位置后，信号就被移除了。

隐藏频谱传播的很多方法都使用了一个名为快速傅里叶变换的数学算法。它使用了一个余弦和正弦函数集合产生一个原始数据模型，这个模型可用很少方法调谐以隐藏一个信号。更现代的变化只使用余弦（离散余弦变换）、正弦（离散正弦变换）或更复杂的波形（离散微波变换）。

14.2.1 范例

图14.3显示了一个以每秒44 000次取样的一个0.33秒的声音文件段的曲线图。声音文件用一个从30.000～－30.000的，按顺序的14 976个整数记录了声音的集中度。我们称之为x_i，其中i的范围是0～14 975。



图14.3 一个0.33秒的声音文件段的曲线图

信息在这个文件中是怎样传播的呢？一个简单的频谱传播方法是抓取一个数据块并且把信息的一小部分加到这个块里的每一个成分中。图14.4显示了这个声音文件里瞬间时刻范围在8200～8600之间的数据曲线图。通过从每一个位置增加或减少一小部分数量，一个信息就被编码。同样的解决方法可以用于视频数据和其他任何能抵制小干扰的的数据源。

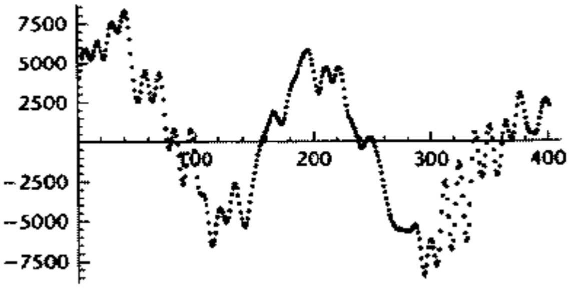


图14.4　图14.3中范围（8200，8600）的一小部分

这里有一些8250到8300的原始数值：2603，2556，2763，3174，3669，4140，4447，4481，4282，3952，3540，3097，2745，2599，2695，2989，3412，3878，4241，4323，4052，3491，2698，1761，867，143，－340，－445，－190，203，575，795，732，392，－172，－913，－1696，－2341，－2665，－2579，－2157，－1505，－729，6，553，792，654，179，－548，－1401，－2213。这个范围（8200，8600）里的数值加起来等于40 813，其平均值为102。

一个基本算法编码1位的步骤是：选择某个强度系数S，然后把所有成分平均值的绝对值安排在S上面（如果信息是一个1）和S下面（如果信息是一个0）。

对这个基本算法来说，S的正确值的选择从某种意义上说是一门艺术，它被块尺寸，真正信号的强度，声音的本性，以及几个其他因素所混淆。我们设$S=10$。如果要被编码的信息是1，那么什么事情都不用做。因为平均值102已经在10上面了。

尽管如此，如果信息是0的话，那么在没有任何错误涉及的情况下，这个平均值就至少需要减小92或者更多。当数值范围在±7500之间时，从每一个成分中减去100就不会使信号失真。当然，有一些成分具有像6或－190的很小的值，它们失真的程度更大，但是这在我们感知的极限下是允许的。

一个更完善的机制把失真成比例地展开。这可以用以下公式计算：

$$x_i = x_i - x_i \times \frac{\text{总变化移动}}{\sum |x_j|}$$

如果这被减小到每一个数值x_i，那么总和将随着总变化移动。

因为数据被展开成一个更大的块，所以这种方法在最小有效位里简单地编码信息方面有几个优势。任何想简单地交换最小有效位的一个随机集合的人将清除最小有效位信息，但是却不会给信息带来任何影响。这些随机变化将抵消掉并且在总和上没有任何合成串音。如果平均值的绝对值在S上方（比S大），那么它求和后仍然不变；如果它在S下面，那么求和后它仍然是在下面。

如果变化被抵消掉，那么随机噪声也应该对信息有一些小的影响。加在某一个位置上的低频干扰将被在另外一个位置上减掉一个低频干扰而互相抵消。当然，这要依赖于噪声如我们所期望的那样运转。如果数据块的尺寸足够大，那么奇数就代表随机噪声将抵消其本身。

除此之外，这个机制就没有其他的弱点。攻击者可以在所有位置插入一些大的数值。改变几个100到30,000的小的成分是一个使平均数失真的方法。这个随机攻击是拙劣的，并且可能因为很多原因而失败。低频干扰是可以被感知的，因为那样很容易被用户发现。当声音文件被反向播放时，它们也可以被消除。很多电子系统都可以移除短的、随机的低频干扰。

当然，也存在很多实践的限制。很多压缩算法为了移除文件的复杂性而只使用很小数量的数值或量子。例如，8位μ-规则编码，对每一个数据成分只使用256种可能的数值。如果一个文件用这个机制进行压缩，那么当每一个成分的数值通过转变成最近量化值而被压缩时，任何用这个技术编码的信息都会消失。

在选择数据块尺寸和它所能携带信息量方面也有很多实际性的问题。如果数据块很小，那么信号将完全有可能清除数据。

例如图14.4的第一部分，显示了成分x8200到x8300之间的数据。几乎所有的成分都在0上（大于0）。总和为374 684，并且平均值为3746.84。从每一个成分中减去这个大数值将使信号显著地失真。

更大的数据块就更有可能包括允许算法工作的足够的信号，但是增加数据块的尺寸会减小可被编码的信息数量。

在实践中，一个有1000个成分或每秒1/44的数据块看来是工作在一个像图14.3所示那样的声音文件中。下面的表格显示了数据块里的平均值。数据块中最小的数值大概是最大的数值的1%，如果S被设在3%左右，那么这个信号要被编码，其问题应该不大。

$x_1 \sim x_{10}$	19.865
$x_{1000} \sim x_{1999}$	175.589
$x_{2000} \sim x_{2999}$	−132.675
$x_{3000} \sim x_{3999}$	−354.728
$x_{4000} \sim x_{4999}$	383.732
$x_{5000} \sim x_{5999}$	−111.475
$x_{6000} \sim x_{6999}$	152.809
$x_{7000} \sim x_{7999}$	−154.128
$x_{8000} \sim x_{8999}$	−59.596
$x_{9000} \sim x_{9999}$	153.62
$x_{10\,000} \sim x_{10\,999}$	−215.226

14.2.2 同步

通过对很多成分加上或减去小数量来储存信息也会受到同步问题的影响，同样的同步问题还会影响水印和信息隐藏算法，但是它却是更富有弹性的。如果有很多重要的成分在文件一开始就消失了，那么同步性的消失就会毁坏信息。

这个机制提供了一个同步损失的逐渐的测量。设想数据块有1000个成分。如果只有一小部分，比方说20个，从文件开始就消失了，那么这个改变就不太可能毁坏信息。把信息展开分配到块的每一个成分保证了仍然还有980个成分携带有信息。很明显，随着不同步数量的增加，信息的质量将减小，当裂缝变到有数据块尺寸的一半时，就会达到一个峰值。我们

注意到错误只出现在数据块中位编码由一个0改变到一个1或由一个1改变到一个0的地方，这是非常有趣的。

同步也可以通过尝试抽取信息而被自动地察觉。设想接收者知道一个隐藏的信息已经在一个声音文件里被编码，但是接收者并不知道信息开始和结束的地方。用一个指南来搜索，找到正确的偏移量并不是很难的。

信息可以包括很多奇偶校验位，一个简单的误码检测方法。就是说，每8位后，一个基于前8位里的数字1和0的额外奇偶校验位就被加到通量上。如果有一个奇数的话，它就被设为1；如果有一个偶数的话，它就被设为0。这个基本的误码检测协议在远程通信中被频繁地使用。

第3章讨论的错误检测和校正代码。

这个机制也可以被用来使信息同步化并且通过一个强力搜索找到数据块开始的地方。文件可以使用一个潜在偏移量的变化得到解码，最好的解决方法是使用最大的正确的奇偶校验位的那种方法。因为信息的质量和一个连续函数很相近，所以搜索从某种意义上说是更智能的。稍微地改变偏移量相应地也只稍微地改变正确的奇偶校验位。

很多像Stirmark校验那样的攻击能毁坏同步。第13章讨论了有关它的一种方法。

也可以使用更复杂的误码校正代码。最好的偏移量需要位通量上要有最少的校正数。它惟一的问题是更多的校正能量需要更多的位，这意味着要尝试更多的潜在偏移量。如果每个字有12位并且每一个位有100取样编码，那么校正偏移量的搜索就必须把0到12 000之间所有的数值都试一遍。

14.2.3 系统加强

通过使用另外一个随机源改变数据从文件中加或减的方法，用多重抽样传播的信息就可以被加强。设α_i是一个系数集合，它可以修改计算总和及信号被抽取的方法。作为计算基本和的替代，是通过系数计算总和的加权值：

$$\Sigma\alpha_i x_i$$

如果α的值是由一个加密安全随机数字源产生，那么它们就可以看做是一个密钥。一个简单的方法是使用一个随机位通量产生α的值，使其等于+1或-1。只有能同样存取随机数字源的人才能计算出正确的总和并抽取信息。

这个过程也限制了一个为了毁坏信息而加入随机低频干扰的攻击者的能力。在第一个例子里，攻击者总是使用正的或负的低频干扰，并且使得总和要么完全是正的要么完全是负的。如果α的值被平等地分配，那么重低频干扰应该被抵消。如果攻击者为了模糊信息而加入越来越多的低频干扰，那么它们将变得越来越可能相互排斥对方。

很明显，通过选择像2，10或1000那样的不同的α的值，复杂性就会增加，但是这种解决方法看起来并没有很多明显的优势。

14.2.4 复合信息分组

像这样的随机数字源也可以用来选择奇特的或不连续的数据块。没有原因可解释为什么从x_1到x_{1000}的成分需要一起作用才能隐藏信息的第一位，从文件中任意选择的1000个成分都可以完成上述任务。如果信息的发送者和接受者访问了由相同的密钥产生的相同的同步随机数字通量，那么他们都可以抽取正确的成分并且在数据块中把它们集合在一起以隐藏信息。

这个方法有一些优点。如果成分是随机选择的，那么数据块的尺寸就可以更小一点。如14.2.1节所述，x_{8200}到x_{8300}之间的数值大部分都是正数，并且平均值也很大。在不使数据块里的数值失真的情况下，要上下调整平均值使其大于或小于*S*值是不可能的。如果成分是相邻的，那么数据块的尺寸就需要很大，以保证它们包括有足够的变化来保持小的平均值。从整个文件里随机选择成分有效地减少了这个问题。

这种方法也允许复合信息被分组。设想Alice通过从声音文件中选择100个成分来编码一个1位信息，并且调整平均值使其大于或小于*S*。现在，设想Bob也通过选择他自己的100个成分组来编码1位信息。这两个人有几次选择可能是同样的成分，但是奇数是非常小的，以致于两个之间没有任何重叠。即使Alice通过增加她的数据块的平均值编码一个1，而且Bob通过减小他的数据块的平均值编码一个1，在两个数据块里有很少成分的情况下，这个工作过程也将不会失真得太大。平均值仍然和原值很接近。

设计一个系统依赖于对奇数的计算，并且这个过程是直接向前的。如果文件里有*N*个成分并且每一个数据块里有*k*个成分，那么在一个数据块里选择一个成分的奇数值是*k/N*。在块和块之间有任何重叠的奇校验可以用经典二项展开式来计算。

如图14.3所示的声音文件里，有14 976个成分。如果数据块尺寸是100个成分，那么从一个块中选择一个成分的机率大约是1/150，选择100个成分并找到无交集的可能性大约是0.51，找到一个交集的可能性大约是0.35，两个交集的可能性大约是0.11，三个交集的可能性大约是0.02，剩下的就可以忽略。

当然，这种解决方法会增加可分组成一个文件的信息的数量，但是它却牺牲了对同步的抵制性。如果数据文件遗失了信息，那么选择不同成分并且把它们集合成一个新的数据块的任何复杂方法都将受到阻碍。如果文件遗失了第一个信息，那么从文件中选择第一个、第189个、第542个以及第1044个成分都会失败。

14.3 块比较

14.2节通过增加平均值使它大于或小于某个*S*值的方法，把信息展开到一个数据块中。如果平均值是可预测的，那么这个解决方法就可以很好地工作。而声音文件里的例子使用接近于零的平均值，为什么如果其他数据通量的平均值在发送者和接受者之前预先知道，它们就不能很好地完成工作呢，这是没有原因可解释的。

另外一个解决方法是调整算法以适应手头的数据。如果文件是图像文件，因为它通常包含具有不同的统计轮廓的信息，所以这个将工作得更好。一张在阴影下的黑暗的照片和一张阳光明媚下的白雪覆盖的山脉的照片在它们各自的文件里都有不同的数字。在这种情况下，预测平均值对发送者和接受者来说都是不可能的。

这里是一个简单的解决方法：

1. 像以前那样把成分分进数据块。它们既可以是相邻的组也可以是由某个加密安全随机数字生成器选择的随机选集。
2. 把数据块分组成对。
3. 计算数据块里成分的平均值。
4. 成对地对比平均值。
5. 如果平均值之间的不同大于某个极限值（$|B_1-B_2|>T$），那么就否决这个对并且忽略它。
6. 如果不同比极限值小，那么保持这个对并且在里面编码信息。设一个对表示一个1，这里$B_1>B_2$。并且设一个对表示0，这里$B_1<B_2$。从专门的成分中加上或减去一个很小的数量以确保信息是正确的。为了加上一些错误抵制，以保证$|B_1-B_2|>S$，这里S是一个信号强度的测量。

广泛地抛弃不同的一对，这里的不同大于T处的不同，有助于确保修改将不会太大。发送者和接受者可以检测到不同的数据块并考虑排除它们。

再说，选择数据块的正确尺寸、极限值T以及信号强度S需要一定的艺术感觉。更大的数据块意味着平均值定律有更多的空间花费在更大的带宽上。T值的减小相应地减小了需要改变的数据数量，其目的是为了以排除更多信息为代价，来编码正确的数值。

14.3.1 最小化量化误差

在一个令人惊讶的细节里，当数据值被紧密地量化时，增加块尺寸的策略就开始失败了。更大的数据块意味着信息可以以更小的变化展开到更多的成分里，但是在一个点确定后，这些点就变得太小以致于不能测量。例如，设想一个最简单的灰度级图像，其中每一像素的数值从0～255。加上一个很小的数，比方说0.25，并不会彻底改变任何像素的值。因为实际上通过很小的加法没有像素能够被改变，所以没有人可以恢复信息。

这个问题可以用这个算法来修正。设S是展开到数据块中所有成分的数，如果这是以相同的等份被展开，加到每一个数据块上的数就将比量子化的数值小。

当S大于零时，重复以下步骤：

1. 随机选择一个成分。
2. 给这个成分增加一个量子。就是说，如果它是一个像一个像素那样简单的、呈直线性的编码数据值，那么就给它加1。如果它是一个像声音文件里的一个成分那样的被对数编码的数值，那么就选择下一个最大量化值。
3. 从S中减去这个数。

数据块中所有成分的平均值将仍然增加，但是只有成分中的一个子集才会改变。

14.4 快速傅里叶解决方法

很多频谱传播解决方法使用了Fourier analysis（傅里叶分析）的一个数学分支，或使用了一个更现代的wavelet analysis（小波分析）理论的修订版。完整的分支是以Jean-Baptiste

Fourier（1768—1830, 法国数学家、物理学家）的著作为基础的，他使用一组正弦和余弦函数产生出了一个模拟化函数的新颖的方法，如图14.5所示。这个解析结果证明它对求解很多微分方程是很有用的，并且它还经常被用来解决工程学、化学和物理学问题。

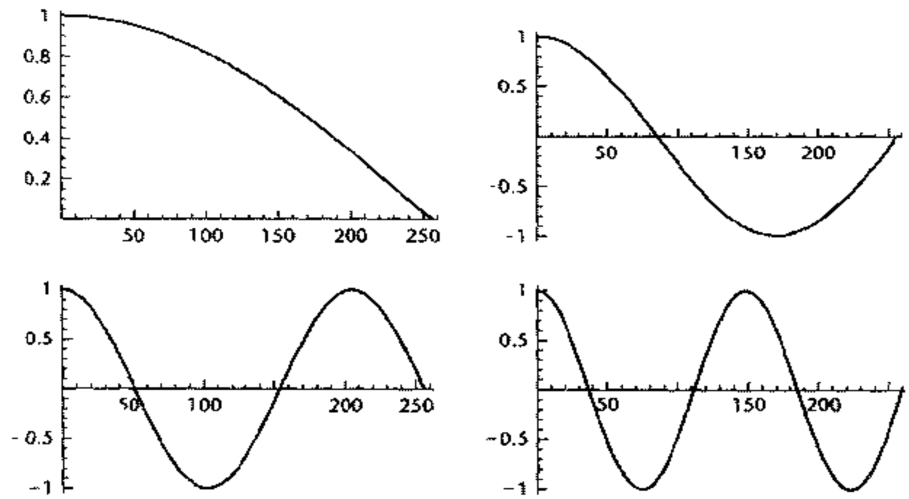


图14.5 使用傅里叶变换的四个基本余弦函数

傅里叶提出的这个机制对密码学也非常有用，这是因为它提供了一个把几个信号一起嵌入到一个更大的信号里的简单方法。大量的奖学金投入到这个主题使得快速地测试理论和发展工具变得更加容易，其最大的一个贡献，即所谓的快速傅里叶变换，是一个今天通常被用来隐藏信息的数字数据文件的最佳化的算法。

其基本想法是采用一个数学函数，f，并且用另外一组函数的加权值来表示它：$\alpha_1 f_1 + \alpha_1 f_1 + \alpha_1 f_1 \ldots$，$f_i$的选择从某种意义上说是一门艺术，并且使用不同的选择解决不同的问题可能会更好。一些最普通的选择是像正弦和余弦那样的基本调和函数。实际上，流行的discrete cosine transform（离散余弦变换）已经在像MP3音乐压缩和MPEG视频压缩那样的函数里被使用了，即使用$f_1 = \cos(\pi x)$，$f_2 = \cos(2\pi x)$，$f_3 = \cos(3\pi x)$，等等。（图14.6显示了作者使用一个离散余弦变换重新编码初始值的持续图。）今天，很多研究都致力于找到能更好地完成特殊任务的更好的和更完善的函数。有关小波的章节（14.7节）指出了一些更普通的选择。

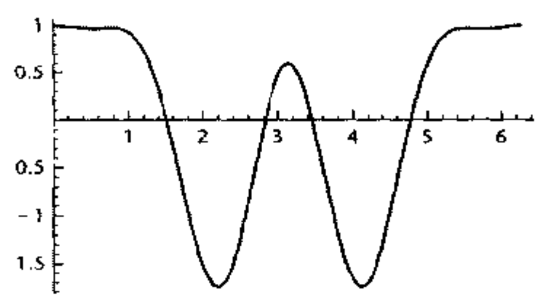


图14.6 作者为一个傅里叶系列重新产生的连续初始值，把图14.5所示的4个函数加在一起：$1.0\cos(x) + 0.5\cos(2x) - 0.8\cos(3x) + 0.3\cos(4x)$

很多以数学为基础的傅里叶分析目的是建立几个特征，使它们对难题是有用的。例如，余弦函数是一个组合，它们是orthogonal（正交的），一个意味着这个组是尽可能有效的专业术语。如果没有函数f_i能被其他函数的总和所代表，那么这组函数就是正交的。就是说，$\{\alpha_1, \alpha_2, \ldots, \alpha_{-1}, \alpha_{+1}, \ldots\}$中没有值存在能使得：

$$f_i = \sum_{i \neq j} \alpha_j f_j.$$

这组余弦函数也形成了一组充分连续函数的基数。也就是说，对所有的连续函数f，存在一组系数，$\{\alpha_1, \alpha_2, \ldots\}$，使得：

$$f_i = \sum \alpha_j f_j.$$

事实上这两组余弦函数都是正交的，一个基数意味着对每一个函数只有一个独一无二的系数选择。在这个例子里，基数必须是无限大的，以代表所有的充分连续函数，但是大多数离散的问题从来都不需如此精确。因为这个原因，关于“充分连续的”意味着什么这个讨论在本书中被省略了。

这两个特征对密码学都非常重要。事实上每一个函数都只能被独一无二的一组数值$\{\alpha_1, \alpha_2, \ldots\}$所代表，这意味着编码器和解码器在工作中都要用到同样的信息。函数形成基础事实上意味着这个算法将能够处理它遇到的任何问题。当然，这组函数的两个要求都可以放宽，就像这本书后面讨论的一样，当然这要在其他步骤提供某些担保的情况下才能做到。

顺便提一下，事实上有很多不同的基础函数，这意味着有很多不同的独一无二的数据表示方法。没有原因可解释为什么基础函数不能在发送者和接受者之间频繁地改变或共享。一个密钥可以被用来选择特殊的基础函数，而且这可以阻止潜在的偷听者的工作[FBS96]。

14.4.1 一些简要的计算

傅里叶分析的基础在于计算，因此在原有的形状中提供一个简要的介绍。如果限制$f(x)$的范围在$0<x<2v$，那么函数f就能用一个无限系列的正弦和余弦表示：

$$f(x) = \frac{c_0}{2} + \sum_{j=-\infty}^{\infty} c_j \sin\left(\frac{j\pi x}{v}\right) + d_j \cos\left(\frac{j\pi x}{v}\right).$$

为计算c_j和d_j的值，傅里叶发展了一个相对比较直接的解决方法，再次用整数来表示：

$$c_j = \frac{1}{v}\int_0^{2v} f(x)\cos\left(\frac{j\pi x}{v}\right)dx \qquad d_j = \frac{1}{v}\int_0^{2v} f(x)\sin\left(\frac{j\pi x}{v}\right)dx$$

事实上这些正交函数是由以下来表达的：

$$\int \cos(i\pi x)\cos(j\pi x)\,dx = 0, \quad \forall i \neq j.$$

如果$i = j$，整数部分就是1。

在过去，对于很多函数f来说，很多这些整数都是不容易被计算出的，并且整个数学分支为了找到结果都在不断地发展。今天，数值积分可以很容易地解决问题。实际上，用数值

分析可以更容易地看到这里的函数分析和向量代数，之间的联系就像是兄弟姐妹之间的关系一样。如果函数f只知道离散的点$\{x_1, x_2, x_3, \ldots, x_n\}$的数值，那么这些对$c_j$和$d_j$的方程式看上去就像点积：

$$c_j = \sum_{i=1}^{n} f(x_i)\cos(j\pi x_i/v) \qquad d_j = \sum_{i=1}^{n} f(x_i)\sin(j\pi x_i/v).$$

由于在数字方式里有如此多的数据被储存和传送，所以离散的方法几乎肯定对现代密码学家来说是更令他们感兴趣的。

14.5 快速傅里叶变换

计算可以是很精确的，但是数字数据是不会参与连续函数的。幸运的是，数学家已经找到了合适计算的方程式版本。事实上，作为快速傅里叶变换（FFT）的离散数据版本是很多用于声音、无线电及图像的数字电子技术的基础。几乎今天所有的多媒体软件都使用了FFT的某些形式来分析数据，找到占统治地位的共扼特性，然后使用这个信息增加数据或压缩数据。音乐家使用以FFT为基础的算法加入混响、阻尼扰动回音、或者在需要录音的地方改变大厅声效。录音公司使用FFT来使音乐数字化并且把它储存在CD（激光唱盘）上。青少年则不断使用FFT来把音乐转变成MP3文件。此外FFT还有很多的用处。

FFT以后的细节超出了本书的范围。这个算法使用一个聪明的数值欺骗程序使乘法的数字最小化，它通常被称为“蝶式算法”。总结了信号中多种频率的强度后，最终结果是一个很长的数值向量。

为了更精确，一个FFT算法接收一个具有n个成分的向量，$\{x_0, x_1, \ldots, x_{n-1}\}$，并且返回了另外一个具有$n$个成分的向量，$\{y_0, y_1, \ldots, y_{n-1}\}$，这里

$$y_s = \frac{1}{\sqrt{n}} \sum_{r=1}^{n} x_r e^{2\pi i(r-1)(s-1)/n}.$$

这个方程式被数学和小波探索程序包所使用，用这个程序在这一节中产生了很多图像。其他的则使用稍微的变换来解决特殊问题。

向量的出现从本质上说是每一个函数跟潜在数据相匹配程度好坏的一个测量。例如，向量中的第4个成分用$\cos(4\times 2\pi x) + i\ \sin(4\times 2\pi x)$测量了有多少图像在普通状态中。

图14.7显示了一个函数$(2+x/64)\sin(4\times 2\pi x/256)$的曲线图。如果这个例子的256个点被加进一个傅里叶变换，那么结果在y_4和y_{251}点就有其最大值。第二个峰值是通过混叠产生的。第一个$n/2$成分里的数值记录从左到右计算的傅里叶变换函数，第二个$n/2$成分从左到右携带了计算傅里叶变换函数的结果。这些结果是对称的。

图14.8显示了应用到图14.7中数据的傅里叶变换的实数和虚数部分。很多物理分析家和电子工程师都使用这些算法来分析无线电现象，像比方说最多的“能量”可以在虚数部分y_4和y_{251}上被找到。一些数学家讨论，图14.8显示了“频率空间”而图14.7显示了“函数空间”。在两种情况下，曲线图都是在测量能被每个成分作为模型的数据量。

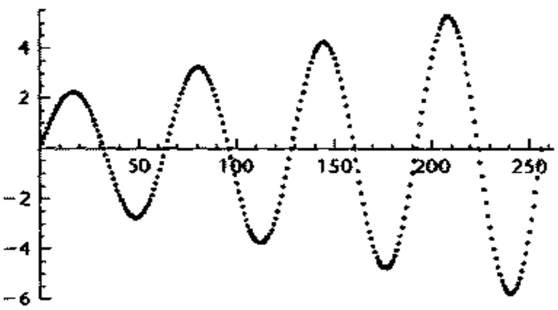


图14.7 显示了256个点，它们是由方程式（2 + x/64）sin(4 × 2πx/256)计算的

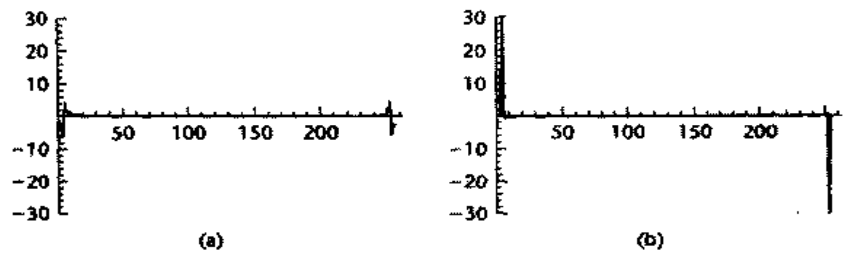


图14.8 这是用图14.7中的数据计算出的傅里叶变换的实数和虚数部分的一个曲线图

由傅里叶变换而来的基本解决方法包括了实数和虚数值。因为这个原因，很多也使用了离散余弦变换（DCT）和它的表亲，即离散正弦变换。它们都有各自的用途，但是当$f(0)\neq 0$时，它们是在$x = 0$点附近做使数据模型化的工作，所以正弦变换便没那么普通。事实上，大多数用户是以他们在终点附近的性能来选择他们的模型化函数的。

图14.9显示了使用于离散余弦函数的一个普通组的第一个4基础函数：

$$\sqrt{\frac{2}{n}}\cos\left(\frac{\pi}{n}\left(k+\frac{1}{2}\right)x\right),\quad k = 0, 1, 2, 3, \ldots$$

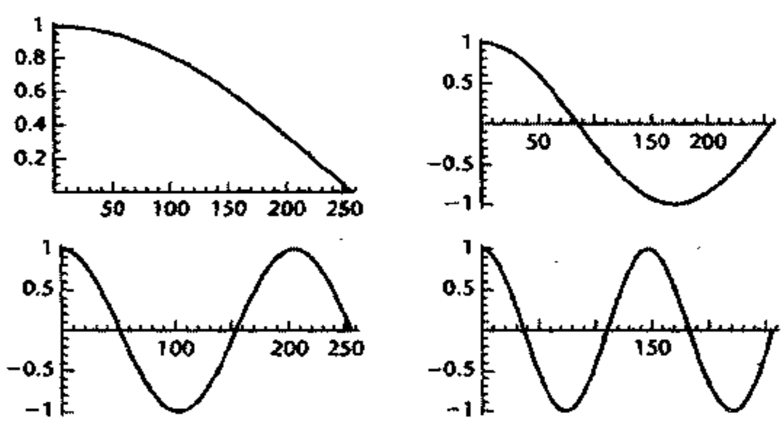


图14.9 在一个流行的数学离散余弦变换应用里的前4个余弦基础函数图

图14.10显示了64种不同的二维余弦函数，它们被用来作为模型化8 × 8块的JPEG和MPEG压缩像素的基本函数。

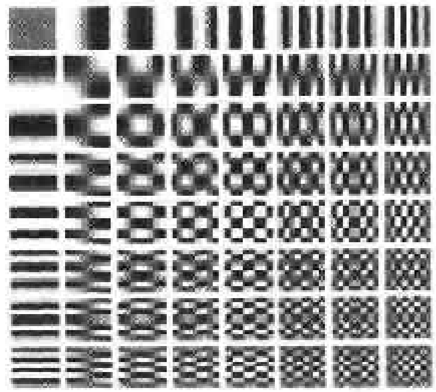


图14.10 在8×8像素格的二维离散余弦变换里使用的64种不同的二维基础函数。每一个特殊点的集中度都暗示了函数的尺寸大小

另一个模型在终点使用了一个不同数值的相似版本：

$$\sqrt{\frac{2}{n}}\cos\left(\frac{k\pi}{n}x\right),\quad k=0,1,2,3,\ldots$$

这些变换每一个都有一个逆操作。因为很多数学操作在“频率空间”里更容易完成，所以这是很有用的。那就是说，数据中每一个频率里的能量数都可由构造这个变换而计算出来。然后，一些基本操作用频率系数来完成，再然后数据就用逆向FFT或DCT被储存起来。

用FFT或DCT的话，消除数据就变成了一个特别容易完成的操作——如果基本信号是反复性的。图14.11显示了在用傅里叶变换消除数据中的4个步骤。第一个曲线图显示了噪声数据，第二个显示了傅里叶系数的绝对值。y_4和y_{251}两个点一点都不受噪声的影响。第三个图显示了所有很小的系数被设为0后的系数值。第4个显示了采用逆向傅里叶变换后的重新构造的数据。自然地，当信号很清晰，并且能被正弦和余弦函数作为模型时，这个解决方法就很有效。如果数据不适合这个格式，那么在大频率和小频率之间通常就有更小的区别，并且移除小频率是很难的。

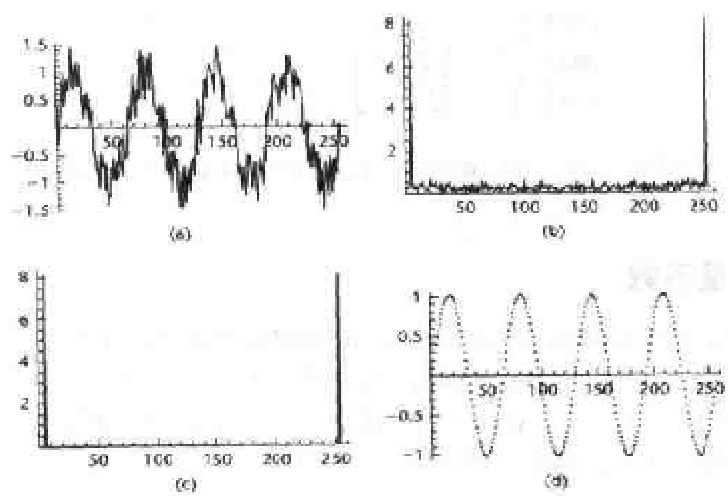


图14.11 用FFT来消除噪声数据的4个步骤：（a）显示数据（b）计算FFT的结果（c）所有小数值被设为0后的数据（d）最后的结果

14.6 用快速傅里叶变换和离散余弦变换隐藏信息

傅里叶变换提供了混合信息和改变系数以隐藏信息的理想的方法，加上一个看上去像cos(4πx)的信号可以只是增加y_4和y_{251}的数值就行了。用一种方法完成它是一个更困难的挑战，这个方法很难察觉和抵制恶意的或偶然的文件变化。

这里有一个简单的例子。图14.12显示了图14.3所示的声音信号的傅里叶变换前600个系数的绝对值。如果需要的话，主频率是很容易确认和改变的。

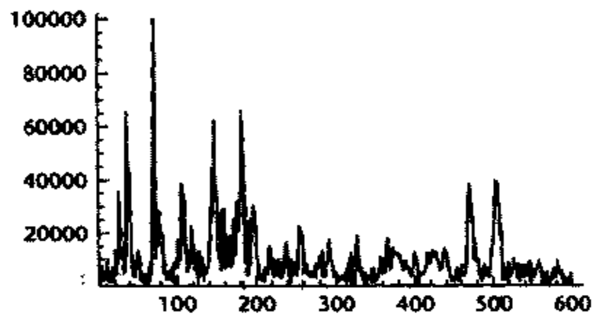


图14.12 在图14.3中所示的声音信号的傅里叶变换前的600个系数

一个简单的水印或信号可通过改变设置y_{300} = 100 000而插入。采取逆向变换后的结果在细节水平上看上去与图14.3相同。数字范围仍然是从－30 000～30 000。它们之间的不同之处虽然很小，但还是可以通过减去被作了水印后的信号的一个原始信号而看出来。图14.13显示了用正确频率在－750～750之间的不同振荡。

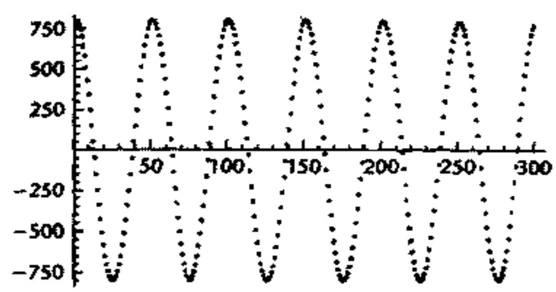


图14.13 从具有一个y_{300}膨胀值的信号里减去如图14.3里所示的原始信号后的结果

14.6.1 调谐大量系数

Ingemar Cox、Joe Kilian、Tom Leighton和Talal Shamoon[CKLS96]提供了一个新颖的方法，就是通过调谐一个数据FFT或DCT的k个最大系数从而在一个图像或声音文件里隐藏信息，把这些称做$\{y_1, y_2, \ldots, y_{k-1}\}$。最大系数对应于数据通量的最重要的部分，它们是有最大“能量”或携带最终图像信息的频率。

Cox和他的同事建议，在最大系数里隐藏信息听起来好像是违反直觉的，但是它却是惟一的选择。乍看起来，在噪声里隐藏信息的最合乎逻辑的地方是在最小系数里，但是这个噪声也最有可能被压缩、打印，或在有漏洞的转化过程中被修改。另外一方面，信号的最重要的部分在没有损坏整个文件的情况下是不可能被毁坏的。

这个方法有很多优点。数据在数值数据或分里被展开。即使有几个成分被改变或删除，信息也可以被恢复。Cox和他的同事证明了携带这个水印的图像即使是在被重复打印和扫描后仍然可以存活下来。当然，其带宽也要比像调谐最小有效位那样的方法的带宽要小。

他们的算法使用了以下步骤：

1. 使用一个DCT或FFT来分析数据。
2. 选择k个最大系数，并且为了简便把它们标记为$\{y_0, y_1, y_2, \ldots, y_{k-1}\}$。更小的系数被忽略。第一部分系数代表通常在这个组里的最小频率，但是它并不受到保证。
   另外一个解决方法是根据系数的视觉意义来安排它们的顺序。图14.14显示了一个Z字形顺序，它被用来选择具有从一个二维变换得来的最低频率的系数。JPEG和MPEG算法使用了这种方法来消除不必要的系数。一些作者建议在这个排序中跳过前l个系数，这是因为它们对图像有很大的影响[PBBC97]。选择下k个系数产生一些选择物，它们对图像描述很重要，但又不能太重要了。

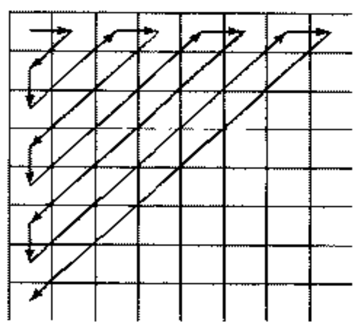


图14.4 在有效位里排序系数的另外一种解决方法。这个图显示了从一个二维变换的系数排序的流行的Z字形方法。左上角的那些图线和最低频率相对应，这样的话就对视觉最有效

3. 产生一个将被隐藏的k单元向量，$\{b_0, b_1, b_2, \ldots, b_{k-1}\}$，这些既可以是简单的位也可以是具有更多丰富信息的实数。在所有情况下这个信息都不会被完整无缺地恢复，所以应该考虑有更多的标识数字，而不是关键位的一个向量。
4. 选择一个系数a，它是用来测量嵌入过程的强度的。这个决定将可能由转接实验和对错误的抵制来做出。更大的数值将对错误有更强的抵制性，但是它们也引入了更多的失真。
5. 通过使用他们建议的这些函数中的一个来修改系数，我们就可以在数据里编码一位向量：
   - $y_i' = y_i + \alpha b_i$
   - $y_i' = y_i(1 + \alpha b_i)$
   - $y_i' = y_i e^{\alpha b_i}$

6. 计算逆向DCT或FFT，以产生最终图像和声音文件。

嵌入数据的存在可以通过逆向步骤测出。这个算法需要原始图像的存在，这是一个在很多情况下严重地限制了它的用途的问题。其步骤如下：

1. 计算DCT或FFT图像。
2. 计算没有嵌入数据的DCT或FFT原始图像。
3. 计算最顶部的k个系数。
4. 使用合适的公式来求出αb_i值。
5. 使用随机成分b_i分布的知识使向量标准化。就是说，如果b_i的值是从一个大约为0.5的标准分布选择出来的实数，那么就可以决定α移动到平均值0.5的数值。通过除法把α从向量中移除。
6. 把向量$\{b_0, b_1, b_2, ..., b_{k-1}\}$和其他已知的向量相比较并且选择最好的匹配。

确认“水印”的最后一步是这个特殊算法的最大限制。任何搜索水印的人必须有一个具有所有已存在的水印的数据库。因为舍入错误即使在图像文件不失真时也会增加不精确性，所以这个算法通常不能辨认出一个完美的匹配。计算DCT和FFT的过程引入了一些舍入错误，把图像密封在一个标准的8位或24位格式里则会增加更多的错误。因为这个原因，我们最好要能得到最合适的匹配。

这使得该算法对某些类型的水印是很好的，但是对隐藏的交流却不够完美（即有一些漏洞）。发送者和接受者都必须在覆盖图像和书本信息代码，或在数据里将被嵌入的水印等方面达成一致意见。

如果这些限制没有影响到你的需要，那么这个算法的确是提供了很多所期望的特性。Cox和他的同事用很多实验测试了这个算法并且证明了它的坚固性。他们用几个256×256像素图像、失真的图像开始测试，然后测试出正确的水印。他们试图把尺寸收缩到一个因式的1/2，使用重JPEG压缩，删除边界范围以外的区域，并且对它发生高频振动。他们甚至打印出图像，影印出图像，并且在不需移除水印的情况下把它扫描回去。在他们所有报告过的实验里，这个算法成功地确认出了水印，虽然失真减小了它的强度。

这个成员组测试了几个著名的攻击，它们可能被一个决定擦除信息的攻击者所装配。第一，他们试图用4个新的水印使图像水印化，最终的测试结果总共有5个图像，虽然不能决定哪个是第一，哪个是最后。第二，他们试图用不同的水印把产生的5个图像平均起来并且发现所有5个图像都可被确认。他们暗示，尽管如此，如果攻击者把它推得更远一点，比方说使用100或100个不同的水印，那么这个算法可能就没有那么坚固了。

14.6.2 把原文从检测过程里消除

保持手头中原始的、不变的数据对像水印那样的应用来说通常是不可接受的。理想地，一个普通用户没有可利用的原始资料也能够从文件中提取信息。很多水印应用假设普通人是不会信任没有被水印过的数据的，这是因为他或她都可能简单地盗用数据。

一个前面的Cox成员小组的算法的变型并不需要原始数据来揭露水印。A. Piva，M. Barni，F. Bartolini和V. Cappellini产生了一个相似的算法，它使用了一种可预测的方法把系数分类到变换式中（参看[PBBC97]）。例如，图14.14显示了一个根据大致的频率给一个二维变换形式里的系数进行排序的Z形状模型。如果这个方法被使用，那么寻找k个最有效的

系数就不必保持手头的原始数据了。很多其他的变型也正在发展中。

14.6.3 回火激活

通过调谐系数插入信息有时对最终图像会有一个很大的影响。图像中最脆弱的部分是平滑的、恒定的面片，它可在比如一个明朗的、蓝色的、无风的夏季的天空的照片里被找到。收听者通常能够听出在一个具有纯音或很长的宁静片段的音频文件里的变化。迅速地改变数据里的数值意味着有很多结构可以用来在里面隐藏。平滑的、缓慢的变化数据意味着没有多余的空间。在这个最抽象的意思里，这是遵循了信息理论的。高平均信息量数据是一个高带宽通道，低平均信息量数据是一个低带宽通道。

一些算法根据伪装数据调整α的值，它们试图通过这种方法使一个水印的强度适应潜在的信息。这意味着水印强度变成了图像里的一个位置函数（$\alpha_{i,j}$）和声音文件里的时间函数（α_t）。

计算α的值有很多方法，但是最简单的才能满足需要。设想一个窗口，它围绕在空间或时间里的一个点，然后从平均数中均衡偏离，是足够简单的（参看[PBBC97]）。对人类感知系统的更复杂的研究，可以提供我们的眼睛和耳朵是怎样对不同的频率做出反应的更深的理解。

14.7 小波

很多算法使用正弦和余弦作为构造数据模型的基础，但是没有原因可解释为什么这个过程应该被限制要单独执行。在最近几年里，研究者们开始用新的能量致力于探索奇特的、不同的函数是怎样产生更好的数据模型的——这是一个研究者们称之为wavelet（小波）的领域。图14.15显示了一个流行的小波函数，即Meyerψ函数。

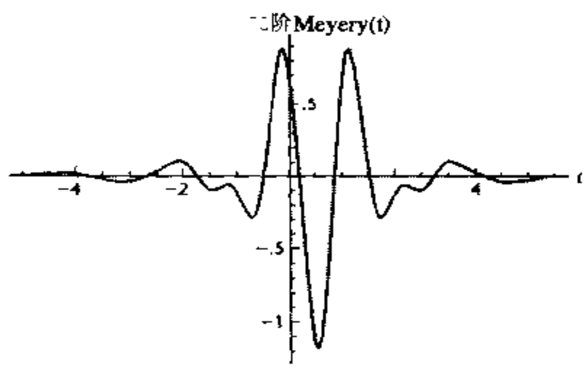


图14.15 Meyerψ函数

小波变换在很多像傅里叶变换或余弦变换中一样，使用同样的方法产生数据模型：它们计算一个特殊函数里有多少像潜在数据那样的系数。就是说，这个计算可以找到相关。尽管如此，大多数小波分析通过改变函数频率和位置或函数非零处的窗口，来给混合增加一个额外的参数。例如，傅里叶变换使用被定义为从$-\infty$到$+\infty$的正弦和余弦。小波变换通过把函数发送到从a到b的一个特殊窗口以外的零值，从而限制了每一个函数的影响。

使用这些定位的函数可以帮助解决当信号在时间或位置上变化时出现的问题。音频文件里的频率包含了音乐或声音随时间的变化，最流行的小波技术中有一个是分析文件的细小部分的。一个音频文件的小波变换首先使用的是被定义在0～2秒之间的小波，然后使用的是被定义在2～4秒的小波，如此类推。结果在第一个窗口里而不是在第一个时间里找到一些频率，那么一些研究者就可以说这个小波变换已经“定位”了信号。

最简单的小波变换就是使用DCT和FFT计算数据的小窗口。把数据分离成更小的窗口是这些算法的一个自然扩展。

更复杂的窗口使用了复合函数，这个复合函数是由在一个被称为**multiresolution analysis**（多解分析）的过程里的复合尺寸定义的。阐述这个过程最容易的方法是举一个实例。设想一个声音文件有16秒长，在第一个传送里，小波变换可以计算整个数据块。在第二个传送里，小波变换可以计算0～7秒和8～15秒的两个数据块。在第三个传送里，小波变换将被应用于0～3、4～7、8～11和12～15的数据块。这是3个阶段的多解分析。很明显，在每一个阶段后把每一个窗口或数据块分成两份是更容易的，但是为什么不能凭空想出复合窗口在复合位置重叠的极复杂的方法呢，这是没有原因可解释的。

多解分析对压缩非常有用，它是和密码学紧密相关的一个主题。一些好的以小波为基础的压缩函数使用了这个基本的循环方法：

1. 使用一个小波变换使窗口中的数据模型化。
2. 找到最大和最有效的系数。
3. 为这些大的系数构造一个逆向的小波变换。
4. 从原型中减去这个模型。所留下的就是不能被小波变换预测的更小的细节。有时这是有效的，有时这是无效的。
5. 如果不同之处小到被感知出无效时，那么就立即停止。否则，就要把窗口分离成很多更小的窗口并且循环地应用同样的过程来消除噪声。

循环和多解分析可以很好地压缩很多图像和声音文件。函数中更大范围的选择意味着压缩可以被调谐至更深一步以便能提取最好的程序。有很多不同的小波可以被利用，并且有一些在压缩某些文件方面要比其他的更好。选择最好的一个通常像是一门艺术似的科学。

在某些情况下，密码学家建议如果只有发送者和接受者知道特殊小波的话，函数的选择也可以通过加入一个密钥来实现。

在这里进入更深的小波领域是不太可能的，这是因为更深的领域更加复杂，并且很多这些复杂性都不会影响隐藏信息的能力。

> *有几个公开的问题是被高度调谐的小波对信息编码或多或少是稳定的。就是说，一个系数里的一个微小变化在被打印和扫描后再集合在一起，那么它还是可靠的吗？换句话说，一个水印强度的α数值必须要多大？*

大多数用DCT和DFT隐藏信息的相同的技术都符合离散小波变换（DWT）。在某些情况下，它们要比基本方法更优越。用DWT找到隐藏的信息比用DCT和DFT找到隐藏的信息更好，因为用DWT可以使以小波为基础的压缩算法更好地存活下来[XBA97]。当然，这意味着一个攻击者可以简单地选择一个不同的压缩算法，或者为了阻止某个文件传播而用很多种方法来压缩它。

14.8 修改

用正弦、余弦或其他小波隐藏信息的基本方法是转变潜在的数据，调谐其系数，然后反向变换。如果系数的选择是很好的，变化的尺寸是容易处理的，那么结果就和原始值非常接近。

在被选择的信息里选择系数和编码数据，上述这个过程中可能会有很多不同的变化。一些更值得注意的变化如下。

14.8.1 确认最佳区域

很多算法都试图分裂一个图像或声音文件，然后确认出隐藏信息的最佳部分。如果系数被改变即便是很小的数量，那么平滑的、稳定的区域就变成杂色的、吵杂的。

多解小波变换是确认这些区域的一个很好的工具，这是因为它们循环地分离图像直到一个足够好的模型被找到为止。平滑的、稳定的部分通常被大规模地模型化，而杂色的、嘈杂的部分则得到成倍时间的分离。自然的解决方法是在模型化最小的、最多细节的区域的系数里隐藏信息。这限制了图像中物体边缘的改变或声音文件里的转变，使得人们更难地确认它们[Are00]。

14.8.2 量化系数以隐藏信息

很多隐藏转换方法通过增加一个水印向量来隐藏信息。通过对比所有可能向量里的系数和选择最好的匹配，信息就被提取了。这对小数量的水印来说是有实践意义的，但是却不适合任意的信息块。

一个更灵活的解决方法是调谐系数以隐藏专门的位。设Q是某个量化因素，一个任意的系数y_i将落在含有某个整数a的范围$aQ \leqslant y_i \leqslant (a+1)Q$。为了编码一位，在最小有效位很小的地方舍去$y_i$的值。例如，如果将要被编码的位是0并且$a=3$，那么设$y_i=(a+1)Q=4Q$[KH98]。

通过找到最近值aQ，任何接受者都可以从y_i中提取一位。如果转变过程是完全准确的，那么将有某个整数使得$aQ=y_i$。如果转换和逆向转换引了一些舍入错误，这种错误是很常见的，那么y_i仍然应该足够接近于某个数值aQ——如果Q足够大。

> Deepa Kunder和Dimitrios Hatzinakos描述了一个以量化为基础的水印，这个水印也提供了篡改检测[KH99]。

Q的值应该谨慎地选择。如果太大，将会导致数值y_i里更大的变化。如果它太小，那么在某些情况下当错误被引入时就很难恢复信息。

这个机制也提供了检测篡改图像或声音文件的能力。如果系数接近于某个数值aQ但又实际上不等于aQ，那么这就是在潜在数据里一些微小变化的结果。如果变化是很小的，那么隐藏的信息仍然可以被提取。在一些情况下，篡改检测是很有用的。一个立体声系统或电视可以用不完整的水印回避反向运行的文件，这是因为它们是某个人试图毁坏水印的证据。当然，它也有可能是某个不完整的拷贝过程的结果。

14.8.3 在相位里隐藏信息

离散傅里叶变换用一个实数和一个假定的数值产生系数。这些复杂的数值也可以在极坐标中被假定为一个幅度或角度。（如果$y_i = a + b_i$，那么$a = m\cos(\theta)$和$b = m\sin(\theta)$，这里m是幅度，θ是角度。）很多DFT的用户感觉到这个变换在角度系数里的改变要比在幅度系数里的改变更敏感[RDB96，Lic]。

这个方法很自然地使用于系数的尺寸。如果它们被旋转了（$\theta + \psi$）度，那么小的数值就被小量地调谐，大的数值被大量地调谐。

这样改变角度需要注意对称性。当一个DFT的输入由实数组成，在加密例子里几乎总是会有这种情况，那么角度就是对称的。这个对称必须被保留以保证实数值将从逆向变换中出现。

设$\theta_{i,j}$代表角度系数（i, j）。如果ψ被加到$\theta_{i,j}$上，那么$-\psi$就必须被加到$\theta_{m-i,n-j}$上，这里（m, n）是图像的维数。

14.9 小结

把信息在一个图像的很多像素或一个声音文件中的单元里展开，这增加了更多的安全性和处理的困难性。把每一位的信息分离成很多块并且把这些块分布于整个文件，这减小了被察觉的机会并且增加了对毁坏的抵制。越多的信息在文件里被展开，就有越多的冗余块攻击。

很多解决方法使用了像傅里叶变换那样的很好理解的软件。因为很多压缩算法都使用同样的工具，所以这些工具在增加水印的技术里通常非常地流行。由于这些算法都使用同样的变换，所以在这些情况下水印通常被压缩而保留。

- **伪装** 通过给文件里的很多数据成分加入一些很小的变化，信息就在文件里传播开来。当所有小的变化被加在一起时，信息就出现了。
- **安全性** 这些变化是很小的并且是分散的，所以它们要比其他方法更安全。每一个分布的信息都能抵制攻击和变化，这是因为一个攻击者必须毁坏足够多的信号才能改变它。信息被隐藏的地方越多，攻击者要确定它的位置并毁坏它就越难。
  通过使用一个密钥来产生一个被加进数据里的伪随机位通量，这个系统就可以变得更安全。只有持有密钥的人才能够移除这个额外的加密系统。
- **怎样使用** 在最抽象的意义里，这个过程是：在具有伪随机位通量的文件里选择很多位置，把数据分离成很多小的部分，以及把这些部分加进所有的位置。通过找到正确的位置，然后把它们加在一起就可以恢复信息。

如果使用了一个快速傅里叶变换的话，这个过程就是非常有效的。在这些情况里，数据可以通过调谐变换后的系数而被隐藏。这可以增加或减小不同频率的数据。

第15章　人工合成的世界

扣篮

现场报道评述员：对蒙大拿州射手队的队员来说，事情看上去有一点困难。他们在一半时间的时候已经有14分的落差了，但是现在爱达荷州步行者队在一个回合里已经投中了两个三分。他们突然在比赛中受到了阻止，射手队叫了一个暂停以便重新安排战术。

Color Man: 用那种方法在记分牌上增添了6分，真正地发出了一个信号。对这些选手来说这很容易办到。

PBPM: 那两个人大喊，"我们仍然在这里。你们没有那么容易就击败我们。我们为我们的毅力和勇气感到自豪。"重点是在勇气上。

CM: 镇静也同样需要。他们在底线和罚球线之间不停地运着球。并且做手势，"我们可以抛开不稳定的状态。我们可以把'弹药'传给你，并且强迫你投篮。我们了解篮球。这也是我们的比赛。"

PBPM: 甚至会有一个令人惊讶的信息潜在意思。我相信步行者队的队员们正在告诉射手队的队员们，比赛过程和比赛结果是不同的。是的，射手队是在主场以22分的优势击败了他们，但那是两个月前的事情了。现在，Jimmy D的腿更好了。他的速度更快了。他的伤是有一次喝酒太多在光滑的水池地板上摔倒弄的，但是现在这一切都过去了。令人厌恶的、使人走入迷途的、以色情诱惑骗取男人金钱的女朋友也成为了历史。步行者队提醒射手队，一点可的松被时间证明是治疗膝盖问题的一个解决方法，但是一生中不停地运动和断绝坏的关系是治疗人的心脏的更好方法。这些就是我认为的在那些三分球里被编码的信息。

CM: 我们现在从暂停中回来。让我们看看射手队做些什么。

PBPM: 现在射手队正在运球。Carter急停然后把球传过半场给Martin。他做了个向左的假动作，然后向右突破。那是一个很宽的通道。他跳起并且砰、砰、砰。这是一个扣蓝。步行者队甚至没有任何防守。

CM: 喔！正是这里发送了一个信息。像刚才那样的一个扣蓝叫到，"你们认为三分球就可以吓倒我吗？你们以为我会在意你们刻板的不过只是自大的跳投吗？球将飞向哪里是没有问题的。没有人站起来摒住呼吸看球是否能进。没有重要的停顿，没有安静的扫视人群，也没有显著的压力。这个球的命运就是球网，而且球网也没有任何问题。"他根本就不是在隐藏什么东西。

15.1 创造的世界

很多声音和图像文件的算法在噪声里围绕隐藏信息而旋转。真实世界的数字化版本通常等待一个信号的额外平均信息量。但是计算机制图学及合成技术的提高，意味着图像和声音是在计算机之中开始一个生命的。它们并不是天生就是真实的世界，并且，所有的本身的平均信息量都是不变地由植物、光、动物、衰变、成长、腐蚀、风、雨，以及我们知道的其他一些东西渗透出来的，而人工合成的世界则是被定义的完美的世界。

乍一看，完美对隐藏信息来说并不好。完全人工合成的图像是以数学开始的，并且这意味着一个数学家能够找到使世界模式化的方程式。一个人工合成的球的图像在光照下有一个没有扭曲的完美的倾斜度，这同样可以在由一个破机器制造的不完美的球的图像里被找到，也可以在一个由过载电子系统提供能量的大量生产球形物照射而得到。

人工合成的世界的规则性，使得加密系统用额外信息确认图像变得很容易。即使是在最小有效位里的很小的变化也可以被察觉。惟一的优点是随着模式复杂性的提高，任何检测过程也逐渐变得越来越复杂。这提供了充足的应用覆盖。更好的计算机制图学技术比任何检测错误的算法都发展得更快，更复杂的模式将会比我们所能推测的来得更快。

如果应用限制并不太好，那么人工合成的世界就代表携带附加的信息。信息可以在人工合成过程中被编码，以代替在最后生成的图像或声音文件里隐藏额外的信息。

隐藏信息有充足的机会。很多计算机制图学算法使用随机数字生成器产生一些不完美位和由它们带来的不真实性，任何这些随机数字通量都可以被“劫持”以携带数据。

另外一个源可以在对人工合成的数据的调谐（tweaking）中找到，也许还可通过改变数据的最小有效位来找到。一个图像版本可以把一个球放在坐标（1414，221）上，另一个版本把它放在坐标（1413，220）上。一个电影的水印版本可以编码某一个字符位置的真正用户名称和他们开始谈话的时刻。每一个电影版本在这些项上都有一些稍微不同的值，只有正确的用户才能抽取这些微妙的变化。

一个更复杂的隐藏信息的位置能够通过改变模型的物理性质而找到。空间的特性很容易轻微地改变，音乐听起来确实是相同的，音乐家可以在同样的时间开始演奏。但是回音的尺寸和特性可能会改变一点点。

> 程序MandelSteg是由Henry Hastur发明的，它是在Mandelbrot組的一个图像的最小有效位里隐藏信息的。这个人工合成的图像计算的精确度达到了7位，而信息则是隐藏在第8位里。参看15.2.2节。

有很多方法可以使整个文件的模型物理特性所使用的参数得到改变，最重要的挑战是要保证这些改变在图像或声音文件里是可察觉的。对其他方法来说可能有些问题，而在这里却一点问题都没有。很多压缩算法通过移除无关信息的调整来节省空间。尽管如此，目标的位置和空间的物理质量都不是无关的，乐器的音质和和录音棚的特性也不是无关的。使这些区别变得模糊的压缩算法应该避免，至少应该被严肃的音乐和图像文件用户所避免。

> Markus Kuhn和Ross Anderson建议调谐发送到监控器的视频信号可以发送信息，这是因为监控器的电子枪发射了很多电磁的“噪声”[KA98]。

设计使用这些技术的加密算法从某种意义上来说是一门艺术。隐藏额外位的地方如此之多，以致于安排给它们的急待解决的问题一定要找到。改变应该要足够大，以便被一个算法所发现；但是改变也应该足够小，以便不被人偶然地发现。

15.2 文字定位编码和OCR

第6章和第8章显示了怎样建立人造的原文记述和怎样在过程中隐藏信息，一个在文本文件里隐藏信息的更简单的技术是打乱字母本身。

其中一个最容易的解决方法是通过变换相近的、但不是相同的字符来编码一个信号。数字“0”和大写字母“O”在基本字形方面是很接近的。数字“1”和字母“L”的小写“l”通常也是很难区分的。第4本信息隐藏著作——以前的会议论文集，其封面就是使用这些相似点来携带一个信息，作者是John McHugh[McH01]。

> Matthew Kwan发明了一个称为“雪花干扰”的程序，它通过添加0～7位之间的任意一位在每一个正文行的末尾隐藏3位信息。

如果字形相似，那么信息就可以通过交换两个模型而被编码。因为OCR程序通常是使用上下文来区分两个信息的，所以要发现已印好的两个版本之间的不同处是很复杂的。如果数字“1”在一个由文字、数字、字符组成的字词（word）的中间被找到，那么程序就会调整其感知到的错误。

如果字形是相同的，那么当数据是在电子表格里时，交换对隐藏信息仍然是有用的。阅读文件时没有人会注意这些不同之处，但是数据将是可推断出的。

电脑黑客的一些成员通常故意含糊地交换相似的字符，使这达到了极限。数字“4”和大写字母“A”具有一些相似点，数字“3”看上去像一个反向的大写字母“E”。这个技术可以逃避关键字的查找和自动文本分析操作程序，至少可以延用到人们的拼音变得标准和非常熟练为止。如此而来，一个有短语“3L33t h4XOR5”的文件就开始看上去可疑了。

霍夫曼也提供了一些基本拼音和语法错误的校正。隐藏的信息有时可以通过引入看起来好像很随意的一个拼音错误而被编码①。

15.2.1 定位

另外一个可能的解决方法是简单地调整字母、字、行和段落的位置。排版工作更像是一门艺术，很多注意力在放在一页中安排这些字母的算法上。当做这个决定的时候，信息可能一直是被隐藏的。

出版本书使用了LATEX和TEX排版系统，它们是通过在一个标点符号后（而不是在一个字后）插入更多的空白处来调整行，美国的排版机通常在标点后放入相当于平常的3倍的空间。而在法国，则避免这个区别并且把它们设为相同。这个机制很容易定制，并且改变跟在任何字符后面的空白处的“拉伸能力”是有可能的。这一段被排版以便字后面的空白处是

①一些人可能会因为溜进文本里的错误而责备校对者，但是，也许这正是在发送一个秘密文件……

以“e”或“r”结尾的，即便它们是标点符号。（TEX指令是sfcode`e = 3000和sfcode`r = 3000）

把整个文件里的这些值改变成更小的值，相对来说是很容易做到的。在很多情况下，发现这些变化也是很简单的。很多商业的OCR程序在一页中会不断地发生一些小错误，但是使用空白处的隐写系统通常更加精确。检测出空白处的尺寸通常要比推断出两个墨水渍之间的不同处更容易一些。

Jack Brassil，Steve Low，Nicholas Maxemchuk和Larry O'Gorman使用了很多不同的技术做实验，并在排版算法里引入了一些很小的变换[BO96，LMBO95，BLMO95，BLMO94]，这使得处理像TEX那样的公开资源工具变得很容易。这些工具还包括很多修改算法的指令。

其中一种最成功的技术是移动原文的个别行。它们显示了整个行都能被成功地向上或向下移动一个或两个6/100英寸。如果有任何歪斜从图像中被消除，那么移动一行就很容易被发觉。只要文档现在和水平线足够接近，那么个别行之间的距离就能用足够的精确度测量出来，以便识别已移位的行。

测量一行的最简单机制是把它“拉平”为一维空间。也就是用墨水在每一行里计算出像素的数字。图15.2展示了图15.1里每一行像素的亮度的求和结果。如果它是完全空白的，那么每一行像素的最大值是255。因为这个原因，第二行就会出现一个比最大值小很多的波谷，这是因为它更短而且它是由更多的空白空间组成的。

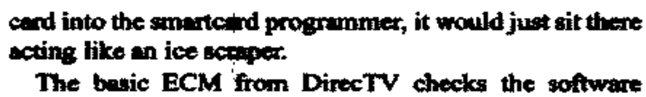

card into the smartcard programmer, it would just sit there
acting like an ice scraper.
The basic ECM from DirecTV checks the software

图15.1　三条已印好的文本行，以每英寸400像素的速度扫描

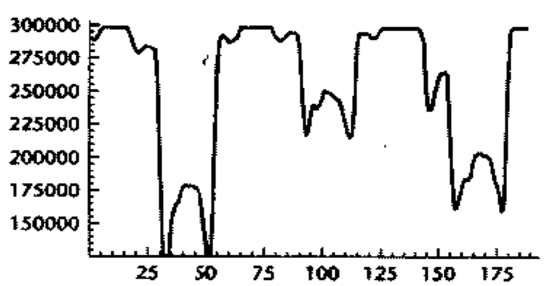


图15.2　图15.1中所有行的一个求和曲线图。空白处通常被分配为255，所以中间的短线不太显著

波峰和波谷的结果也会因为字的不同而不同。第三行有更多的大写字母，所以在对应的大写字母的水平线开端，有更多的显著的波谷。字形的选择也会改变这个曲线图的形状，事实上，这个曲线图就是被用来确认字形的[WH94]。一般来说，确认有衬线的字体的基线（baselines）比没有衬线的字体的基线更加容易，但也并不总是这种情况。

通过把专用字向上或向下转变3/100英寸，更多的位就可以打包进每一行。因为这样使得用来测量水平基线转变的信息更少了，所以检测过程变得更加复杂。短字没有长字那样引

人注意。在实践中，我们通常把多样的短字组合在一起，并把它们转变成一个块（block）。

Jack Brassil和Larry O'Gorman的实验显示了基线的位置，以及任何在基线里被编码的信息都可以从文本图像中有规则地取出，甚至是在图像被反复地复印后也能做到。向上或向下移动个别行，一个或两个6/100英寸的变化通常是能被察觉出来的。他们注解说他们的结果需要一个全定向文件，这个文件的文本基线和光栅线排列在一起。推测起来，一个更完善的算法可以弥补由于模仿图形保真而带来的错误，但是把纸张在第一位置准确地排列起来可能更加简单。

15.2.2 MandelSteg和秘密

任何图像都是隐藏信息的一个选择物，但是有一些图像要比其他一些图像好。通常，有充足变量的图像好像非常完美。如果相邻的像素具有不同的颜色，那么人眼就不能察觉出个别像素里一些微妙变化。这个概念导致Henry Hastur产生了一个程序，这个程序能翻转一个Mandelbrot组的最小有效位。这些图像在数学领域里非常流行和广泛。这个程序以MandelSteg而著名，它对Cryherpunks存档的源代码是很有用的（ftp://ftp.csua.berkeley.edu/pub/cypherpunks/steganography）。

MandelSteg指南注释到，这个系统有几个弱点：第一，某些人可以简单地运行数据恢复程序GifExtract来移除位。虽然有几种不同的设置，但是一次只能有一种工作。因为这个原因，作者建议使用Stealth——一个从PGP（Programmable Graphics Processor，可编程图形处理机，是一个程序）信息中去掉成帧文本，只留下噪声的程序。

还有一些其他的弱点。Mandelbrot图像作为数据的一次装填，不管你使用什么编码方法，只要某个人能找到密钥数据的一个模型，那么数据就可以被取出。虽然Mandelbrot组看上去非常随机和混杂，但是却有很多结构。在一个简单的方程式（$f(z) = z^2 + c$）收敛之前，每一个像素都代表重复数字。毗邻像素的像素数值通常是不同的，但是它们仍然与它们的普通生成式方程相连接。因为这个原因，认为研究一个不规则图像的最高有效位和确定位的位置是有可能的。这样一来，就允许某些人重新计算最小有效位和抽取答案①。

15.3 回波隐藏

虽然在声音文件的噪声里隐藏信息是一种很好的解决方法，但是信息可以通过好的压缩算法被擦除。Daniel Gruhl、Anthony Lu和Walter Bender建议调谐（tweaking）空间的基本声音以隐藏信息[GB98]。而这仍然可以被足够强大的压缩所隐藏，它通常更有可能抵制标准算法。回音是录音的一部分，熟练的收听者的耳朵是经过专门训练的，所以他们能够察觉它们中的很小的变化。好的录音师和他们的压缩法在努力提供尽可能的逼真性来试图消除回音。

很多录音软件程序已经包括了从一个录音中加（或减）回音的能力，它们能通过改变回音的强度和消失的速度来改变回音的特性。

①David Joyce在互联网的网站上提供了一个Mandelbrot图像生成器（http://aleph0.clarku.edu/djoyce/julia/explorer.html）。

信息也包括改变的强度和衰变的长度。Gruhl、Lu和Bender通过改变在回音开始之前的时间长度，成功地编码了一个单独的位。1得到一个短的等待（大约0.001秒），0得到一个稍微长的等待（大约0.0013秒）。通过把信号分离成多段并在每一段编码一个位，我们就可以编码大于1的位。

信号通过自相关（autocorrelation）被检测。如果音频信号用$f(t)$表示，那么位就可以通过计算$f(t+0.001)$和$f(t+0.0013)$而被检出。那些携带信号“1”的部分一般将产生一个$f(t+0.001)$的更高值，那些携带信号“0”的部分将产生一个$f(t+0.0013)$的更高值。

可利用的带宽取决于取样速率和一个较少的音频文件数。更高频率的声音和更高的取样速率能把信号分成更短的片断以提供精确的结果，两者在组合里是独立的。Gruhl，Lu和Bender成功地使信号分成的各段持续1/16秒的时间。

在很大程度上，这个算法的成功依赖于人耳的聆听。一些人天生就有很好的听力，一些人则训练自己的耳朵以便获得更好的听力，还有一些人则两者皆有（既天生有好的听力，又训练过自己的耳朵）。录音行业继续使用像回波隐藏这样的技术进行实验，以便在录音中加人一个水印。结果通常是非常好的，但是仍然是有问题的。在很多情况下，普通人是不能察觉到附加的回音的。那些能察觉出附加回音的人，很多都认为声音更丰富了。但是，一些该行业中的最好的艺术家仍然拒绝在他们的完美的音乐上做任何改变。有时，这个争论会带有一些讽刺。波段存放区是用来制造噪声的，回授丰富的音乐有时候会被抱怨加人微小回音位作为水印。看来这个过程仍然需要一个艺术家的继续涉人。

15.4 小结

停止移动文本中的行或停止添加回音是没有理由的（它们应该不停地进行）。任何人工文件都可以在建立过程中被调谐（tweaked）。真正的难题是在建立检测算法上，这种算法要能够发现并取出文件中的任何变化。在某些情况下，数据是很容易被利用的。例如，在Macromedia的动画格式里逐步展开的一个动画图像可以在位置和时间选项里编码信息。通过使用关于文件格式的公共分配的信息，这个数据就很容易从文件中被取出。

如果数据不能从文件中取出，这时很多图像和声音的人工智能分析技术就变得很有用处了。例如机器视觉算法，它能够抽取一部电影里动画特征的位置和方位。音效师可以使用回波检测和消除工具帮助确定携带隐藏信息的回音位置。

- **伪装** 任何人工文件，例如文本文件、声音、光，或者可能有一天包括嗅觉都可以通过在合成中调谐一些参数来携带信息。
- **安全性** 在很大程度上，它的安全性依赖于文件的本身特性和艺术家的技术。高质量的图像在很多地方都有一些轻微的扭曲，这使得它们可以携带几位的信息而不被任何人发现。
- **怎样使用** 任何在信号中使用加法或调谐最小有效位的隐写技术，都可以被指导人工合成的原始数据所使用。在很大程度上，所有这些有关隐写术的技术都被应用在过程中的一些较早的步骤。例如，一个动画师可以选择一个特性部件的位置，然后把这个数据提供给绘制机，在动画师选择位置和绘制进行之前，隐写术就完成了。

第16章 水 印

用水印标识人们的专利

为什么水印被限制在数字文件和纸片上？有什么方法可以像水印一样留下痕迹吗？我们有一个主张，要为“通过幽默笑话嵌入一个思想模仿水印”提供一个专利申请。

主张：

1. 在一个试验者中通过以下方法来嵌入和观察水印：
   - 从一组打趣的话、闹着玩的挑逗话、谜语、双关语、打油诗、笑话、滑稽顺口溜、模仿作品和讽刺文学选出并确定一个可笑的句子。
   - 提供说过的可笑的句子来说试验者的一种设计好的方法，以吸引他们的注意和说过的笑话句子在他们头脑的灌输，并且创造一个思想模仿水印的例子。
   - 通过重复说那句说过的笑话来观察面前的说过的模仿水印，以分析那些既笑过又声称那些笑话他们已听过的人对试验者的反应。
2. 主张（1）的方法，不停地重复已说过的笑话，直到它深入被试验者的神经模式中。
3. 主张（2）的方法，一次又一次的重复已说过的笑话，增加反应的时间和反应的程度，以此来增加对水印的观察。
4. 主张（3）的方法，多次重复地说原来的笑话语句，增加对试验者足够的敏感性来促使他们掐住下一个重复笑话的人的脖子，增加对水印的进一步可能的观察。

16.1 嵌入所有权信息

对于算法应用的一个最主要的要求就是隐藏被保护的版权信息。这个工作要求以某种问题来隐藏文件所有者的信息，以及确定隐藏后阻止被以未授权的形式使用。这是个很高的要求，因为出版业希望有数字化的书、音乐、电影和其他的多媒体。把电脑弄成一个系统，以为这样有的生产者可以像他们想像中那样用许多古怪的机器来像造钱一样验证。有的建议放弃前面的n－1章的破坏秘诀，而去控告那些被确认了的破坏者的最后一个。另一些则建议应该惩罚那些向朋友推荐复制品的人。所有的这些手段都取决于某些安全复制保护的方法，很多的想法包括隐藏信息和隐写术。

用隐藏的信息来保护文章、音乐、电影和艺术通常被称为“水印”。一个生产厂商的明亮的图像被印在说明书的纸上，这就形成了水印。因为隐写术能够隐藏文件作者的信息，所以在生产时期就已在信息中讲清楚是哪个人什么时候可以使用它。确切地说，电脑显示的文件可以正确地了解隐藏的信息和被作者所做的正确事情。

按生产者的要求来处理文件并不是一件容易的事。本书中所有的算法都可以隐藏任何复杂的说明。例如，对带隐藏信息的文件什么可以做，什么不可以做。一些复制保护只用了70位，这个数目也适合大多数的文件。

仅仅插入信息不是最好的方法，因为水印要面对不同的威胁。许多标准的隐写术系统为了防止被发现而采用了混合方法来避免侦测。水印也是用来隐藏的，但不是用来被发现的。许多消费者和伪造者很快就会知道水印是怎样的，如果他们想复制的话。真正的挑战是阻止消费者或伪造者复制和转移水印。

这可不是一件简单的事情。理想的水印可以保持在文件里，即使是被编辑、减切、压缩、旋转或任何其他形式的变形。可惜的是，没有一种水印可以做到，尽管有许多种方法用来抵抗基本的变形。

由Ingemar J. Cox，Matthew L. Miller和Jeffrey A. Bloom创作的*Digital Watermarking*是这一快速发展领域的很好的概述[CMB01]。

防止基本的复制是很容易的。文件的数字化复制是很精确的，可以带上任何一种水印。但不是每一种复制都是精确的，艺术家就常常减切和旋转图像。用压缩算法变成声音或图像文件来增加被复制的灵敏度，也只是一部分重要的信息可以做到。伪造者找出复制的显著信息而把隐藏信息放到后面。实际上，防范所有的威胁是不可能的。

这并不令人惊讶。复制一个文件就意味着复制所有的人所能感觉得到的东西。如果在文件中天空是明亮的蓝色，那么复制品也应该是明亮的蓝色天空。如果文件中有铃声，那么在复制品中也应该有音质差不多的铃声。但是，如果文件中的某部分没有被发现，那么复制品就不可能复制到那一部分了。

水印制造者面临着一个艰巨的难题。非常好的隐藏信息是不为人类感知的，所以很容易被复制品隐藏在后面。对大多数隐写术来说，最好的技术对水印是不可防守的。压缩的算法和不准确的复制品都是可以去除水印的。

但是人类容易看到和听到的信息不是有价值的艺术，对音乐和图像的变形能毁灭和伤害那些在特别产业中不可接受的复制品在市场上的质量。

如果不能够发明理想的水印，就不能解决实际中的一些问题。如Digimarc发明的一种嵌入水印中的工具Adobe Photoshop。这种软件可以插入数字标签到图片中以此保护所有者的权力。这种方法，或许是对Photoshop的讽刺，在第14章所提到的译码技术中，可以对抗许多基本的变形和引进的改变。这种技术并不是完美的，最初的测试表明在图像模糊和清晰之间旋转45°将会破坏水印。

所有的水印方法都面临着这样那样的技术缺点。计算对抗的总数是很困难的。从一张图像的弯曲、折叠、损伤的基本变形情况来看水印的使用期，这个收集是个好的开始，但是一系列的变形几乎是无止境的，而且很难做出模型或者解释。

以此看来，水印发明者还要考虑科学的界线和试着确定出阻止不同的威胁。什么该做，什么不该做。朝这个方向他们创造出一类水印来描述不同的类型和它们的用途。下面的部分举出了不同评价的几种类型。

16.1.1 易碎性

有些水印只要改变一幅图像就被改变了。把信息隐藏在最小有效位（参见第9章）是不坚固的水印，因为在这种强度上交换位不可能恢复所有的信息，甚至造成错误修改和增加多余的东西。

尽管易碎性水印不常使用。一些人建议在嵌入水印后，一有干预水印就立即会被破坏。如在水印中包括一些文件的数字标签，用此来保证文件没有被改动过。

16.1.2 连续性

一些水印随着变化的增大，抵抗变形的范围会逐渐消失。越来越大的变形产生越来越弱的效果，这就是现行的水印。

从14章我们知道，连贯性通常在微小的译码方法中被描述。水印本身是一种描写图像参数的媒介，图像中少量的改变就会引起参数的少数改变。一个与算法匹配的媒介通过发现最好的匹配来找到水印。

许多情况下，水印的长度取决于水印本身的信息数量。区别大量水印要求不同水印间有一小段距离。一小段距离意味着只有一点点变形就可以把一种水印变成另外一种水印。

16.1.3 水印尺寸

信息有多少“位”是最好的呢？一些水印简单的隐藏信息的位，算出文件中存储的位是很容易的。

另一些水印技术不是隐藏在单个位中的。它们通过一种方法增加变形：用变形文件的形状和位置来指示文件的拥有者。隐藏大量信息意味着有许多不同的有区别的变形图案。在某些情况下，打包不同形状的图案并不容易。因为文件隐藏处的尺寸、形状和相互作用都不容易模仿或描述。

16.1.4 隐蔽探测

某些水印要求给探测器提供一些额外的数据，这可以是没有被作过水印处理的原始图像或声音文件，或者还可能是密钥。最好的提供隐蔽探测的方法，就是提供尽可能少的信息给寻找一种水印的算法。理想探测器将会检查文件、核对水印，并加强对水印携带信息的约束。

许多好的保护技术都要求有隐蔽检测功能，提供一种干净的，未被水印过的复制品给电脑使用者来使他的目的落空，但这并不是说没有隐蔽的技术是没有价值的。有一些方法，设想水印只是为了获得事实后的东西，如证据。一种方法就是把ID合法所有权的数目嵌入到一个水印文件中，如果文件后来出现在公开媒体上，或者是因特网上，则文件的拥有者可以用一种不隐蔽的技术来索取文件水印和追寻文件的源头。它们仍可以保持复制品未发表前的原样。

16.1.5 对复合水印的阻碍

存储更多的隐藏信息是一种对隐藏有信息的文件最容易的攻击。使用相同的算法常常确保相同的隐藏点将会改变以携带新的信息。

一种理想的水印可以在插入的数据部分携带多种消息，不需要调整就可以纠正它。第9章的一些最小有效位设计提供了用密钥选择隐藏数据的技术。许多可以携带消息而不出问题，特别是纠错后的东西偶尔会发生冲突。

不幸的是，这种理想的方法通常由于是脆弱的，以及其他一些原因而不合需要。因此，有另外一种折衷的办法，即把信息放入水印中，减少了另一个随机水印对它进行破坏的可能性，但是它也增加了“只要改变一点就会损坏”的可能性。

16.1.6 准确性

许多水印设想通过牺牲准确性而达到对失真有一定的强健性，许多是依靠发现最好的、可能的匹配，因此，如果失真足够大的话，错误的匹配是危险的。这些算法以一种奇异的方法牺牲准确性。一些变化甚至产生正确的答案，但太大的变化能产生戏剧性的错误答案。

16.1.7 保真性

测量水印最难的就是在水印制作过程中有许多失真的引入。一些看不见或听不见的失真可能是合乎需要的，但它们容易被压缩算法当做是所有不需要的数据移除出去。

最好的失真引入技术要足够小，以便让大多数观察者所忽略。他们通过改变重要细节的长度和位置关系来获得成功，一个传统的方法就是通过微妙地改变录音室里的声音。听觉通常不能准确分辨出8×8的房子和20×20的房子的声音有何不同。在某些地方这种方法也会失败，艺术家就能正确地调整声音效果而不需要干扰多数听众的听觉。

第14章介绍了许多小波编码的技术，如成功地通过改变最大系数组合来描述图片的相关长度等。最小的参数很容易被忽略或除去，如果不扭曲图像以致失真，那最大的（系数）就不能被移动。方法就是改变相关长度直到它们符合某种图案。

16.1.8 对帧的阻碍

水印的一种潜在用途就是确定合法拥有者与复制品的不同。泄露、伪造文件的人通常想除去自己的名字。一种最简单的技术就是把多样复制品合起来再平均它们。如果一个像素来自一个版本而另一个像素来自于另一个版本，那么这是两个水印都能幸存的好机会，或者这个新水印会“陷害”一个无辜的人。有一些方法试图避开这种攻击，它通过嵌入有签名的复制品或制作确认码，这样来延长水印的寿命。

16.1.9 密钥

读取水印需要密钥吗？插入水印需要密钥吗？一些算法用密钥来控制防止未经授权的人插入数据来伪造文件或制造假水印。另一些人用密钥来保证只有授权的人才可以读取水印和收集信息。在第12章已描述过这种方法。

没有什么算法能提供这些特性的完美结合。在某些程度上说，没有一种方法不是通过牺牲一个特性来完成另一个特征的。好消息就是水印可以通过失去一个而得到另一个。

16.2 基础水印

下面包含了近年来最基本的几种水印的方法，它们简要地描述了在现实工作产生的版本中实行的技术。

1. 从文件开始是一种标准形式。

2. 选择一种方法分解文件的重要部分。一种简单的方法是用余弦分离法，用某些参数与余弦相乘模仿信号，例如设$\{C_0, \ldots, C_n\}$是参数。
3. 参数表示不同的组成部分的大小。理想的情况是，这个例子将表示大的参数在文件中的作用大，小的参数作用小。如果是这样，那么用一种方法把小的参数去掉，它们不重要，而且它们可能会随文件本身的一点儿改变而被完全改变。
4. 通过要找近似值替代而使系数量子化。为C_i量子化的一种简单技术就是找一个整数K_i，如果K_iQ近似于x，Q就叫做"量子"。在有些式子中，取对数或指数也可以达到目的。
5. 把｛b_0，…，b_n｝做成水印。每扭转一个系数就插入一个水印，让C_i位于Q的奇数与偶数之间，那就变成了$K_iQ \leqslant C_i \leqslant (K_{i+1})Q$。对系数$C_i$编码$b_i = 0$，就把$C_i$设为$Q$的偶倍数；编码$b_i = 1$，就把$c_i$设为Q的奇倍数。
6. 从$\{C_0, \ldots, C_n\}$的新值中用反向转换的方法重建原始文件，如果分解的过程清晰就是正确的，所做的改变不会完全改变图像。当然，一些试验表明，Q值的作用是必要的。

这种水印译码技术可用在不同的例子中。如拆解图像和声音文件最重要的就是要建好Q，当Q比较大时，抗干扰能力就强，Q值的作用可以在这点上体现。通过取出水印的检测算法可以看到这个值：

1. 为了取出水印，开始可以通过同样的用一系列系数$\{C_0', \ldots, C_n'\}$模仿文件的解结构（译者：deconstructive）技术来得到。
2. 至少要找到$k_iQ - c_i'$的整数k_i。如果没有干扰项，那么$c_i' = k_iQ$。如果文件模仿得好的话，文件改变一点，系数相应也会改变一点。少量改变的结果同样都是显现k_i的值。
3. 如果k_i是奇数，那么$b_i = 1$；如果是偶数，那么$b_i = 0$。这就是水印。

许多水印技术提供了一种附加保护层，即在水印位中包含一些误码校正位，｛$b_0, \ldots, b_n$｝（见第3章）。另一种方法就是比较已知的一套水印，然后发现最好的匹配。在某种程度上讲，这些都是相同的技术。选择Q的尺寸和误码校正的数量，以使你确定充裕的、可用的数量。

深度地依靠分解算法是个很好的方法。离散余弦转换也是好的方法，但有缺点，即细微的旋转就会在系数中产生很大的变化。一些研究者用旋转极抵抗这种特性，它在文件的所有方位产生相同的系数。尽管这种方法常常由于文件被裁切过，文件中心点已改变而被破坏。每种方法都有长处和缺点。

16.2.1 选择系数

另一个任务就是选择改变系数，一些人建议改变大的和显著的部分。水印发展初期就是这样的，Ingemar Cox，Joe Kilian，Tom Leighton和Talal Shamoom等建议选择改变一幅图的余弦分离中的最大系数。规格大小决定了这些系数比小的部分更有决定性。浓缩图中的这部分，使它更加可能在压缩和改变中幸存[CKLS96]。

因为感性的原因，另一些人建议集中一系列系数，选择余弦分离法系数中正确的部分对剪切进行抵抗。例如：对于函数$\cos(2\pi x)$，当$\cos(2\pi x/1000)$重复1000次时，$\cos(2\pi x)$重复一次。水印用较小、较短的波比用大的更可能抵抗剪切。在创建水印时，这些短波也产生小的、更有局部性的失真。

16.3 平均值水印

剪裁是水印创作者面临的问题之一。艺术家频繁借用图片和剪裁图片，要做这些艺术家的水印设计，就要承受得住剪裁。

一种简单的方法就是遍及图片重复水印许多次。本章中最先介绍的方法在一些范围做到了，就是使用分解技术，像使用离散余弦转换一样来实现这种方法。许多相同的系数在剪裁后还会常常出现。

这个例子就是用一种很普通的方法来重复水印。这种方法并不难，也不复杂。但是简单的方法往往也有一些优点。

水印由一组m×n的小整数组成。为了简单，我们就把它设定为4×4，赋值｛-2，-1，0，1，2｝。有5^{16}=152 587 890 625种可能的水印，尽管这不能实际地说出两者的不同。以$w_{i,j}$来表示，其中$0<i<m$，$0<j<n$，此时$m=n=4$。

把图像分成4×4块插入水印中，然后把它们的值增加到像素中。像素$p_{i,j}$由$p_{i,j}+w_{i \bmod m, j \bmod n}$代替。如果值太大或太小，用最大的值或0代替，通常是255和0。

怎样恢复水印呢？办法是求像素的平均值。使$w'_{a,b}$等于所有的像素的平均值。$p_{i,j}$中的$i \bmod m=a$，$j \bmod n=b$。在4×4例中$w'_{0,1}$是像素的平均数。如$p_{0,1}$，$p_{4,1}$，$p_{8,1}$，$p_{0,5}$，$p_{4,5}$，$p_{8,9}$等。

接下来的步骤是保证图像的图案分成小块不规则地进入$m\times n$的水印中。那是因为，所有的像素的平均值将是一样的。一张明亮的图片可能就具有较高的平均值，而一张暗色图片可能就会是低的平均值。理想的情况下，这组数会平衡输出。如果平均数是S，那么目标就是找出匹配$w'-S$的、最好的水印。

如果图像没有被剪裁或改变，这还是很容易做到的。如果不是相同的，$w'-S$将应该近似，就可以插入水印。只有当水印值增加到像素或像素值溢出时，才会不准确。

如果图像被正确的剪裁，要恢复水印也是很简单的。如果$m\times n$像素的新边界距离原来的边界是完整的倍数，那么w'和w的值还是仍能正确地排成行。

这种凑巧的事是很少发生的，任何mn种可能的方向都会发生。一种简单的方法就是从已知的水印数据库中比较并恢复图像的值。这就要做kmn个步骤，这里，k是已知水印的数量。如果系统只是用少量水印，这还不是问题。但如果k变大了，那就很难找到了。

另一种方法就是为水印矩阵创建一个规范的秩序，把p和q作规范的偏移，目的就是要找出一对值与p和q对应，这样我们就可以为水印找到同样的秩序了，而不管是什么方案。这是一种低效率的方法，下面是另外的几种方法：

1. 例如，设$z_{i,j}=5^{4i+j}$，5是水印的可能值。
2. 设$F(w, p, q)=\sum z_{i,j}(2+w_{(i+p) \bmod 4, (j+q) \bmod 4})$。
3. 试图找出$F(w, p, q)$所有可能的数值，并选出最大数（或最小数），这就是w的规范的秩序。

如果mn是合理的值，那么它可以产生更多的辨别力在一个数据库中存储每个水印的mn样本。如果它是大的，那么规范的秩序就要具有更多的辨别能力。

将有516种可能的水印不能使用，这一点应该清楚。它们中有的有同样规范的值，每个水印都有15种可能相似的情况。另外的会因为图案在水印里面而变成相同的。

16.3.1 失真的影响

这种水印的成功度取决于对任何改变能取得平均协调性。如果噪声和干扰在图像中是均匀分布的，那么改变将应该是保持平衡的。平衡将能抵消改变。

不是所有的干扰都是一样的。几种“Stirmark”的改变从图像的中间产生或删除像素中的行或列，这可能会完全改变平均数和破坏这种水印。虽然它可以通过简单选择图像得以恢复，但这个过程既不安全又复杂。

16.4 小结

对隐写术来说，水印不是一件简单的事情。许多探究这个领域的研究者都有极高的认识水平，并且想要他们的水印能在经过一系列的干扰和改变后仍可保持完整。

本章中讲了两种基本的演算方法来抵抗一些基本的干扰。考虑到所有的水印，这个讨论避免被卷入许多复杂的决定因素中。如果图像没被改过，找出水印是很容易的。在编排后寻找它就成为一个广泛推测的练习。只有当一种向量足够地近似另一种的时候吗？哪一种才是最匹配的水印呢？这是工程设计的问题，在本章不作回答。要找到最好的方法要建立一个系统，并且用一组简单的图像和声音文件来测试。其他问题则要取决于系数。

- **伪装** 水印是一种隐写术方法，要求用来抵抗大多数的来自聪明使用者和秘密伪造者的攻击。理想的水印将以这样的方法来嵌入信息；只有一种方法可以破坏它，那就是产生足够多的噪声和干扰，以致原图像再也不可能使用或确认出来。
- **安全性** 没有一种事是近乎完美的，但总有一些是可用的。像Digimarc的系统由于有Adobe公司的合作而得以广泛应用。当它们被图谋攻击时，它们可以抵抗不规则的变形。成功的几率取决于个人的演算法和样品文件。
- **怎样使用** 找一张图像，做一些改变，接着试着追捕想侵犯你版权的人。实际上，它的合法性和逻辑性比制作水印工作中的问题还要让人头痛。如果你真

受挫了，那你就像一些公司那样控告他人吧。

第17章 密码分析

对字进行编码

介绍信的作者们往往加一个有含义的秘密层隐藏在那些有事实根据的信件的表面之下。

- 他所有的同事都震惊于他谨慎细致地注意细节和他共享这个信息的热情。
- 让我们说他的意志力就是他的弱点。
- 大胆的文字和彻底全面的陈述是他的朋友。
- 我们总是很惊讶他锤炼论文、杂志文章，以及在实验室里一个很短的讨论的才干。
- 这个候选人是一个精通于不表达出他所包含意思的艺术的专家。他很快就败倒在假情报和周密的诽谤话语里。

17.1 寻找隐藏信息

本书里的许多技术距离完美还相差甚远。一个聪明的攻击者可以通过仔细寻找隐藏信息过程中一些轻微的人为痕迹而确认那些藏有信息的文件。在一些情况里，这些测试可以简单到足以使用高度可靠的自动操作。这个领域通常被称为“密码分析”，一个模仿密码分析（即关于破解代码的研究）的术语。

这个密码分析领域通常更关心于简单地确认一个信息的存在，而不是实际地解读它。这是很自然的，因为隐藏信息这个领域的目标就是隐藏一个存在的信息，而不是给它加密。在密码分析里很多基本的测试就是确认一个信息存在的可能性。恢复隐藏数据通常超过了测试的能力范围，因为很多算法都是使用简单的加密安全随机数字生成器来给插入的信息加密的。在一些情况下，隐藏的位传播于整个文件。有一些算法不能告诉你隐藏的信息在什么地方，但是却可以告诉你隐藏的位可能在哪里。

确认一个隐藏信息的存在对一个攻击者来说通常已经足够了。这些信息通常是脆弱的，一个攻击者实际上不阅读信息就可以毁坏它。一些数据可以通过在存放它的地方储存其他的信息而被擦除。另一些则可以通过翻转一个最低（有效）位随机数字而被确认无效。很多小的、简单的失真能够毁坏信息，此后信息又都是以小的、简单的失真形式而存在的，这样攻击者和接收者都阅读不到信息。

所有的这些攻击依赖于确认一个音频或视频文件的一些特征部分，这些部分是通过隐藏的数据来改变的——那就是说，要找到密码学在效仿或者掩饰方面经常失败的地方。在许多情况下，隐藏的信息比它所替代的数据更加随机，并且这个额外的“完美”通常是很醒目的。例如，很多图像文件的最小有效位并不是随机的。在一些情况下，照相机传感器是不完美的，一些文件压缩造成了随机性的缺乏。可以用一个更随机的（即，更高平均信息量）隐藏信息来代替最小有效位，以移除这些人工因素。

这也有其局限性。很多这些技术都必须调整以攻击特殊软件程序的输出。在一个熟练的操作者手里，它们可以被用来寻找由已知算法产生的隐藏信息，但是当这些方法遇到了来自于稍微不同的算法的影响时，它们就可能开始失败。不能保证密码分析算法都能自动地延伸到每一个软件版本里，没有什么反密码分析的灵丹妙药。

但是也不能保证任何密码分析算法都能抵受住聪明的密码分析。本书里没有一种算法能提供摆脱统计的或计算的人工因素的任何数学保证。如果不小心使用，很多自动密码程序就会引入错误。如果不谨慎使用，很多错误就不那么容易被察觉。

17.2 典型方法

很多关于这个课题的早期工作是由Sushil Jajodia，Neil Johnson，Andreas Pfitzmann，Niels Provos和Andreas Westfeld完成的[WP99，JJ98a，JJ98b]，但是新的工作正在出现。基本算法可以分为几个种类：

- **视觉或听力攻击** 用一种使一个人更容易找到视觉反常的方法，可以使某些攻击能去除图像的有效部分。一个普通的测试显示了一个图像的最小有效位，因为不完美的照相机、扫描仪和其他数字化设施在最小有效位里留下了大结构的回波，所以，完全随机噪声通常能揭示一个隐藏的信息的存在。

  大脑也有能力辨别出声音里的微小不同，很多声音水印生成器都是由于敏感的人耳能够辨别出不同而受到了重击，为了增强信号部分而进行的预处理文件使得他们的工作更加容易。
- **结构攻击** 因为包含有隐藏的信息，所以数据文件的格式是经常改变的，这些改变在数据结构里可以归结为一个容易察觉的模式。在一些情况里，隐写程序使用稍微不同的文件格式版本，并且把它们泄露出去。在其他一些情况里，隐写程序用另一种方法加上文件或额外信息。
- **统计攻击** 像素模型和它们的最小有效位通常能揭示出一个存在于统计轮廓里的隐藏信息，新的数据并没有像标准数据那样拥有同样的统计轮廓。

当然，没有理由限制这些方法。每一个密码解决方法使用某个模型给信息编码，为了使每一种算法都与结构相匹配，可以产生复杂计算方法。如果算法在像素对的相对层隐藏信息，那么攻击就可以计算那些成对的统计轮廓。如果一个算法按照某些元素的顺序来隐藏信息，那么攻击就可以检查统计轮廓的顺序。

不管是密码分析还是预密码分析，在理论上都没有理想的模型。这很大程度上是因为理论模型总是能力范围有限，第7章和第8章所描述的仿真函数在理论上是很难破坏的。Kurt Gödel，Alan Turing和其他人的经典著作坚定地证实一个计算机程序不能有效地分析另外一个计算机的程序。

这样的理论保证是令人鼓舞的，但是它们并没有像听起来那么强大。可以这样解释亚伯拉罕·林肯的话：你可以在一段时间内愚弄所有的计算机程序，并且在所有时间里愚弄某些计算机程序，但是你却不能在所有的时间里愚弄所有的计算机程序。即使在所有时间里没有毁损仿真函数的程序产生，为什么某些东西不能察觉出在许多时候发生的不完整性，这是没有道理的。例如，在棒球比赛画外音里的隐藏信息的软件将在某个时间过后就变得总是重

复，某些统计分析的模式可以揭示出一些什么。它不会在所有时间里都工作，但是它将在某些时段里工作。

密码学家能做得最好的就是不断地改变参数及用来隐藏信息的位置，密码分析家做得最好的就是不断地探索错误遗留的微小模式。

17.3 视觉攻击

最简单的密码分析形式是用人眼或人耳检查图像或声音文件。我们的感官通常能够进行复杂的、直觉的分析，这在很多方面都超过了计算机的能力。如果密码算法不太好，那么首先应该是外观上的改变，隐藏信息想瞒过人的眼睛是第一个挑战。一些基本算法可能出错并且在隐藏信息的过程里改变了大量的颜色，但是大多数都应该产生一个函数相同的图像或声音文件。

计算机即使增加一位，也能够很快地让我们看到一个隐藏信息。如果图像的大多数重要部分被去除，肉眼通常能够毫无困难地认出被编码的信息。图17.1和图17.2展示了一个图像的最小有效位在用EzStego程序隐藏信息之前和隐藏信息之后的图像。

这个图形阐述了图像的最小有效位通常不是任意的。注意到图像的饱和区域在最小有效位里仍然是可见的。当图像是全白的或是全黑的时候，最小有效位就不是随机的（是一个特殊值）。它通常被标记为一个0或一个1。即使是不完全饱和的部分也不是随机的。一个图像里的目标位置和光亮度保证了在颜色上有逐渐的变化。这个递减率也不是随意的。然后就只剩下纯不完美性。数字照相机不可能总是有24位的饱满的敏感度，软件可以抽空24位的图像，但是这只能发生在加上了计算机结果和额外的细节以后。当传感器没有足够的分辨率的时候，最小有效位就不是总是随机分配了。

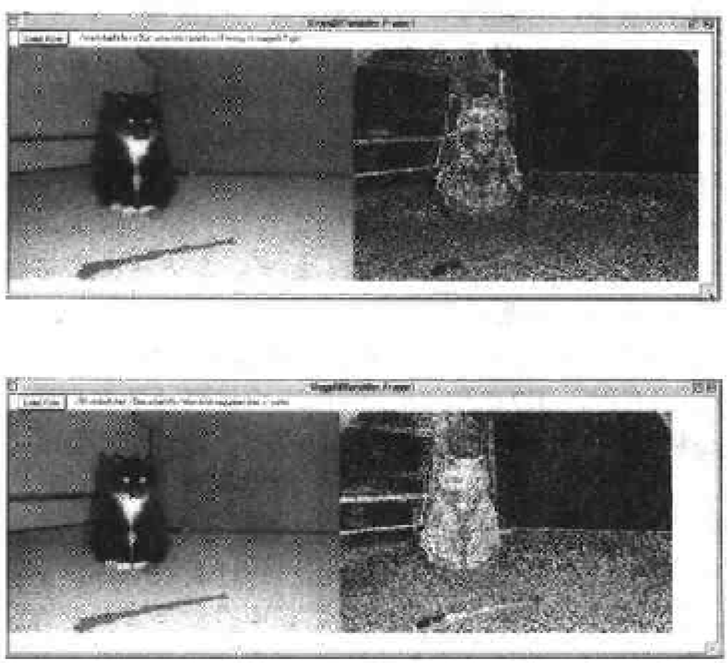


图17.1 顶端的窗口显示了一张照片和用EzStego在图像里隐藏大约6000字节之前的照片的最小有效位。这幅图像可容纳15 000字节。底下的窗口显示了图像及其最小有效位被编码后的情形

在一些情况下，文件压缩算法能使大区域具有同样的最小有效位。例如，JPEG算法储存每一个图像作为某些余弦函数的加权和。如果8×8块里的64像素足够相似，那么算法就将储存平均值。为了节省空间，GIF算法也用同样的值代替相似的颜色。这两个结果都促成防止了最小有效位是真正随机的。

在信息被隐藏于最小有效位之后，所有这些区域都变得更加随机了。肉眼通常是确认这些变化的最快的工具，很容易看到图17.1和图17.2的结果。

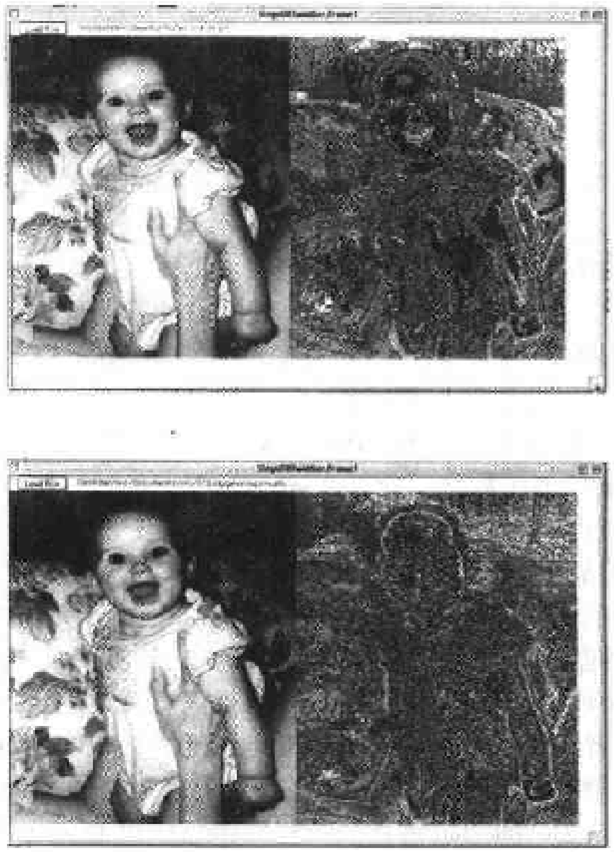


图17.2 顶端的窗口显示了一张照片和用EzStego在图像里隐藏大约17 000字节之前的照片的最小有效位。这幅图像可容纳20 000字节。底下的窗口显示了图像及其最小有效位被编码后的情形

为什么没有产生更复杂的视觉呈现，这是没有原因可解释的。信息并不一定要在最小有效位里被隐藏，这一部分是因为信息通常是脆弱的[SY98]。更复杂的呈现可以混合几个位平面，并且允许攻击者确认出额外信息被隐藏的地方。

17.3.1 听觉攻击

熟练的声音技师和音乐家通常能听出被普通人耳忽略的变化，很多音乐系统的水印创建者发现在大多数人都听不到的区域插入额外的回音是很简单的。尽管如此，被训练过的人耳还是可以立即察觉出它们。

如果数据被标准化，那么普通人耳也能够拾取信息。大多数技术都是依靠人类的大脑：拾取最重音的频率而忽略具有同样频率的但比较温柔的版本。

17.4 结构攻击

在很多情况下，隐写算法超过了数据一个特征结构，这就像一个普通人的书法和一个艺术家的绘画一样，是有区别的。如果你知道你要找的是什么，就可以很快地发现一些算法的结果。

很多基本密码解决方法是在图像的最小有效位里隐藏信息，这些方法被用来代表图像的数据格式所阻碍。当每一个像素代表24位时（红、绿、蓝的数量分别为8位），在最小有效位里隐藏信息是简单的。扫描仪和照相机通常在其被分配到每一个像素的3个最小有效位里留出足够的随机性，以便使储存信息是可行的。

不幸的是，大多数图像并不把24位都分配到每一个图像。像GIF或PGN这样的文件格式通过建立一个选择颜色的调色板来分配8个位或者更少。一个i位调色板意味着2^i种可能的颜色。这能有效地减小图像的尺寸，尤其是当它们和长运行编码（译者：run-length）混合在一起压缩确定同样的像素作较长伸展的时候。

有些情况下，隐藏数据的软件结构和标准结构是不同的。很多GIF文件是用256种颜色记录编写的，即使是只需要很少数量。软件SysCop只在图像里写出（将主存储器中的数据复制到后备存储器中）颜色。额外的效率可以缩短文件，但是它和输出是有区别的。

在现在流行的很多数据包里都有很多低频干扰和一些不兼容性。例如，Hide and Seek（捉迷藏）4.1版本需要所有图像都为320×480像素，StegoDos使用320×200像素图像。这些限制都很容易固定，但是它们显示了快速和恶意的密码软件可能有多危险[JJ98a]。

在压缩的GIF或PGN格式里通过调整最小有效位来隐藏信息，这种方法通常会失败，这是因为调色板人口（译者：entry）之间没有足够地接近。人口01001001可以代表深蓝色，而人口01001000可以表示热辣的粉红色。旋转移动最后一位将使图像歪曲。

第9章的很多加密方法试图通过建立一个特殊调色板来避免这个问题。这一过程会留下一个逃避不了的标记。

一种简单的技术就是选择一个更小的调色板，然后加上一个能用来隐藏信息的复制颜色。例如，S-Tools首先产生一个最佳的32色图像调色板，然后产生7种与红、绿、蓝成分里的每一个不同像素的近复制颜色。如果32种被选择的颜色中有一种有RGB轮廓（64，250，120），那么7种额外的颜色将与RGB值（64，250，121），（64，251，120），（64，251，121），（65，250，120），（65，250，121），（65，251，120）和（65，251，121）相加。这种方法的更旧版本也是共同的。例如，MandelSteg把调色板的尺寸减小到了128种不同的颜色，然后产生每一种颜色的一个复制。

这个方法把3个额外信息位隐藏在每一个像素里——一个有效载荷——但它是一个以一定成本的有效载荷。像这样的调色板是很容易察觉的。当颜色群出现在调色板里的时候，它们通常作为位调整方法的指示器。为了把减少颜色数量过程中的错误降到最小，普通调色板形成算法试图选择尽可能宽的分散的颜色。

另外一些情况，这些算法试图排列调色板的元素顺序，以便一个接一个地放置相似的颜色。因为一个相似的颜色应该是邻近的，所以交换最小有效位应该不会使图像歪曲得太多。EzStego，一个由Romana Machado编写的程序，就是使用这个技术获得了成功。

13.6节描述了怎样以颜色的顺序来隐藏信息，这是在GifShuffle里使用的一个技术。

如果分类方法是公开化的或者很容易算出，那么攻击者就能够窃取位。如果数据由额外的加密层保护，那么信息就不能从随机噪声中辨别出来。

攻击者可以通过检查位的统计轮廓来发现一个存在的信息。一个被加密的隐藏信息应该以相等的概率0和1出现。如果0和1的数量是相等的，那么奇数点就代表一个隐藏的信息。

有两种方法可以阻碍这些攻击。第一种方法是使用一些分类程序加密版本，以防止偷听者集合分类调色板。调色板分类算法并没有确定性，因为它试图在三维空间里安排点以便使相邻两点的距离是最小的。这是“旅行推销商”（一个很难的问题）的离散版本。EzStego使用一个合理的大约值进行猜测。一个加密安全密钥的随机数字生成器代替了使用一个随机源做决定的位置，并且产生了一个密钥分类算法。

第二种方法是使用第6章描述的统计仿真函数，产生一个最小有效位的统计等价模型。原图像的最小有效位模型可以被取出然后用来给数据编码。这种方法可以愚弄一些统计测试，但是不能愚弄所有的测试。在17.3节里描述的一些视觉测试里它就会失败。

17.5 统计攻击

很多数学统计研究都致力于确定一些现象是否是随机发生的。科学家们使用这些工具来决定他们的理论是否能很好地解释这个现象，因为隐藏的信息通常比它所替代的信息更加随机，所以很多这些统计工具也可以被用来确认图像和有一个隐藏文件的音乐。被加密的信息通常是接近于随机的，除非它已经被重新处理过并加上了统计不规则性。

察觉随机性的最简单的统计测试是χ^2（**chi-squared**，希腊字母的第22个字χ的平方）测试，这个测试是求平方偏差之和。设$\{e_0, e_1, \ldots\}$是一个事件发生顺序的时间排列。在这种情况下，它可能是一个最小有效位为0或1的时间数。设$E(e_i)$是取样真正随机条件下的事件应该发生时间数。取样里随机性的数量是用以下方程式来衡量的：

$$\chi^2 = \sum \frac{(e_i - E(e_i))^2}{E(e_i)}$$

高分暗示一个非随机条件——这个条件可能是由一组不完美的传感器产生的原图像或声音文件的一部分。低分则暗示了一个高度的随机性——一些通常和加密的隐藏信息连接在一起的性质。

χ^2测试可以应用于文件的任何部分。通过两个事件的情形，可以分析最小有效位，当最小有效位为0时是事件e_0，当最小有效位为1时是事件e_1。低分意味着位以接近于相等的概率出现，而高分则意味着一个位的数目超过了另外位的数目。在这种情况下，$E(e_0) = E(e_1) = 0.5$。

考虑到相邻最小有效位的模式，一个更好的解决方法是产生4个事件（参看表17.1）。普通图像通常把相邻位设为同样的数值，而有隐藏信息的文件就有不同的相邻位。

表17.1 把两位转变为事件

事件	位	相邻位
e_0	0	0
e_1	0	1
e_2	1	0
e_3	1	1

有很多隐藏信息的文件在这个χ^2测试中通常得到的是低分。更普通的、没有掺杂的图像通常能得到高分，如图17.1和图17.2所示。

Neil Johnson，Sushil Jajodia，J. Fridrich，Rui Du和Meng Long报导说，测量相近的颜色数目是一个很好的统计测试方法，能发现在最小有效位里有隐藏数据的图像。一对相近的颜色在每一个红、绿、蓝成分里不会有超过一个单位的不同，普通建立的文件没有具有额外插入数据的文件那么多的相近颜色对。如果图像用一个像JPEG那样的损耗压缩机制在同一时间储存，则这是真实的。对相近颜色对的数目进行的测试是一个很好的指示器[JJ98a，JJ98b，FDL，Mae98]。

这些测试通常很好地确认了基本最小有效位的隐写术。尽管如此，更复杂的隐藏信息机制将避免这个简单的测试，并且需要一个人工调谐的算法。设想发送者正在隐藏信息，方法是选择像素对，并且偶尔交换像素对以编码一个0或一个1。整体的颜色分配和它们的最小有效位在算法过程中不会被改变。交换并不是指改变统计轮廓的最小有效位。

13.6节描述了怎样以像素的分类顺序隐藏信息。

在某些情况下，如果攻击者能够认出颜色对，那么一个测试的增强版本就能确认出一个隐藏位集合。一个有隐藏信息的图像里的颜色对，其顺序应该以相等的频率出现，而那些普通图像里的颜色对顺序则应该可能有一些不完美性。

更完善的测试可以被调谐以用于不同的应用。程序JSteg是通过改变JPEG算法里使用的整系数最小有效位来隐藏信息的。在标准照片里，较小的系数比较大的系数更普通。数值1的普遍程度是数值2的两倍，而数值2的普遍程度是数值3的两倍，这样依次类推下去[Wes01]。当这些值的最小有效位被调整以隐藏信息时，出现机率就被平等化。1和2的数目变得相等，3和4的出现率变得相等，如此等等。如果两个系数只有它们的最小有效位不同，那么它们的出现率随着信息被隐藏而变得相等。

χ^2测试可以帮助确认JPEG照片中系数很相似地出现的地方。

17.5.1 全面的统计

另外一种解决方法是检查由关于图像的一组函数的应用所产生统计表。Hany Farid注意到，很多用于图像模型的小波函数通常产生有区别的统计轮廓[RC95，Sha93，BS99]。他在多种规模里应用了一组*quadrature mirror filters*（QMF，积分对称过滤函数），并且发现在某些情况下，由这些小波衰变产生的基本系数统计轮廓能够预测出一个信息的存在或不存在。就是说，平均值、偏差、偏斜和峰值都有足够的不同以致于它们可以区分开来。

像JSteg和EzStego这样的基本程序里的信息可以用接近于98%的准确率被察觉，而像Outguess那样的更谨慎的程序里的信息，其被察觉率只有77%。当然，成功很大程度上是取决于能用于编码信息的图像的尺寸。当隐藏信息大约有负载图像尺寸的5%时，其成功率就会很高。如果图像尺寸下降，那么成功率就会下降到接近没有（实际上有2%）。

一些成功毫无疑问归功于这样的事实：一个像Outguess这样的程序仅仅致力于试图平衡第一顺序统计值。具有更复杂统计值的通用换算衰减仍然受到这些平衡调整的影响。一个被设计用以保持QFM统计值平衡的更完善的Outguess版本可能击败它（不受平衡调整的影响）。当然，保持信息足够小是一个最简单的解决方法。

17.6 小结

对每一种密码算法，都可能有一个密码分析形式来探测它。很多早期算法是相对容易探测的，这是因为它们舍去了统计不规则性。最新的算法则更加的安全，但是只有时间才能告诉我们它们是否能够抵制全面的分析。

一个公开的问题是，是否有一种能够调整统计种类变化的固定描述。简单地交换最小有效位将改变一个静态的特殊组，并使另一特殊组不受改变。尽管如此，这个工作还是需要一个更深层的理解：数字照相机、扫描仪和麦克风是怎样把我们的世界转变成数字的。

同时，由隐写术攻击引起的滑动漂移，其最简单的解决方法就是使信息的尺寸最小化。信息越小，文件中的改变就越少，统计值的歪曲就越微小。在这个领域里，指导规则和在其他领域里一样，就是“不要太贪婪”。

Mark Ettinger使用猫和老鼠的游戏理论比喻隐藏者和攻击者之间的关系[Ett98]。

- **伪装** 因为混合信息所做的努力通常会留下其他的标记，所以数据仿真在隐藏一个已经存在的信息时通常是彻底失败的。而在密码分析方面所做的努力则可以检测出这些模式并且揭示信息的存在。
- **安全性** 很多早期的软件数据包在图像噪声里隐藏信息是很容易被察觉的。图像的结构并不是我们能用数学轻易地描述的。在很多情况下，照相机或扫描仪不会在最小有效位里产生具有真正噪声的文件，这就意味着任何隐藏信息所做的努力将会功归一溃。
- **怎样使用** 最好的密码分析，其研究目标是专用算法和它们留下的不规则统计值。最好的解决方法是了解当时有哪一个软件正在被使用，然后分析它的特征痕迹。更普通的解决方法有欠于精确，而最一般的解决方法是检查最小有效位的随机性。太多的随机性很可能是隐写术的一个标记——或者说是一个好的照相机的标记。

总　结

当写本书的时候，我正受到一些困扰，可能是这些算法某些固有的麻烦和错误。如果罪犯可以如此有效地隐藏信息，那么法律制裁就变得更加困难了。当罪犯犯罪前，警察几乎什么都不能做，而犯罪后证据又很少。本书的所有方法，无论所用的比喻或者聪明玩笑是如何地富有哲理或精于修饰，它们都携带这种隐藏的威胁。

这种威胁在2001年9月11日世贸中心被破坏后变得更为明显。一些新闻报导提供了假设，攻击者可能通过图片隐藏的信息调整他们的行动计划。在我改写这些这一章的时候还没有这些说法的证据，但毋庸置疑它确实发生了。

美国联邦调查局或者至少它的职员明确地主张，需要有权使用所有的通信。如果某人在谈论它、写它、发有关它的邮件，或者传真它，FBI就有可能进行侦听以便他们能够破案。这是个明智的态度。更多的信息只能确定法律制裁是公平和正直的，人们只能通过自己的招供而被判定有罪——而不是联邦密探为了正确判决而指证的口供。

证据对于FBI和警方就像权力一样是抽象的和轶事的。当然，权力经过适当的过程还是可以驯服的。如果证明被存档，且一系列证据都保持原样，权力的滥用就可以减少到最小程度。即使警察比一般的公民正直得多，他们将仍是恶棍警官，被迫进入这个国家所有人的通信之中。这是强有力的工具，这个权力带来的腐败是世界上最古老的问题。

所有的这些情况包括在一个故事里（在我开始出版这本书的第一版之前），这个故事是从一个新奥尔良女人看出窗外，并见到警官殴打她儿子的朋友而开始的。她向国家事务部告发了这名警官。到了午餐时间，警官审问了她那个警官的名字以便进行指控，还有她的住址，她的穿着。他根据她的申述记录了一些文字，但当天晚上，她死了。

我们如何知道这件事的发生？我们如何知道它不是个街道暴力的突发事件？新奥尔良的联合专家跟踪这名警官并窃听了他的电话。他是一个嫌疑犯，是一个为毒品交易保驾护航的恶棍警官。他们记录下了窃听得到的情况。

这有一点疑问，就是安全通信将会使这种情况不能解决。如果除了杀人者没人知道杀人的情况，那么就不会有事，也不会有法律制裁了。

同样有一点问题就是国家安全事务部可以防止这起谋杀。这个漏洞恐怕是来自同僚，但恶棍警官可以监听国家事务部门的电话。这个情形是可以想像得到的。谋杀可以被延迟，直到情况不利于这名警官。面对法院，我们原告的权利不可能意味着是不可能永远地保持被告相同的秘密。

什么才是正确的方法？全部开放可以阻止很多犯罪，但它会遇到其他形式的欺骗和诡计。全部保密保护了一些人，但也给罪犯披上了一层外衣。

在过去，包括FBI和别的部门在内的法律强制机构建议了一个系统，这是个以“密钥契约”闻名的可行性妥协。这个契约系统广播一份加密的对话密钥的拷贝数据包，只能让指定的人浏览。尽管司法部门官方描述了大量的有关密钥和进入的控制，我仍然不能确信它没有漏洞。如果这个工具对街道的警察有用，他们将需要尽快取得密钥。检查记录就只披露他或

她的电话被非法窃听的控诉。但你如何知道你的电话被窃听了呢？除非你发现某人手上的磁带。

一种情况就是用公钥加密会话密钥。只有政府才有权访问它。

这样对这个问题用技术提供最终解决方法就真地没有用了。某些方面，法律强制机构必须给官方参与窃听以破解犯罪。更多的这种能力集中在手上的一个小数字，更强大的是它变得与腐败联系在一起来破坏法律。美国自身拥有许多危险的秘密，如制造原子武器的技术，也受到其中的威胁。对不入流的罪犯在整个国家实现小规模的腐败也没有一点疑问吗？

这种腐败的深度和复杂性导致与期望完全相反的遭遇。**Robert Hanssen**，一个联邦调查局密探，从事反间谍工作，却同时也变成是俄罗斯的间谍。在**2001**年**9**月，美国国防情报局（**DIA**）驻古巴部门的首脑被逮捕，被指控为古巴做间谍。如果目的是为了保护美国的信息不被俄罗斯和古巴知道，我们必须意识到**FBI**和**DIA**手中的窃听技术也被国外势力所掌握了。

如果普遍地使用窃听不是好的对策，那么同意无限制的隐写术和加密术也不是一个好的对策。本书里描述的技术提供了躲避警方搜索的几种方法。加密文件看上去像是秘密，而且秘密可以被认为有罪。尽管第5次修订的美国宪法给了人们否认有罪的权力，但是这有点疑问就是使用这个权力能让人产生怀疑。

> *但在最后他们看到什么或认为他们看到什么并没有关系。终端都安装有全息扫描器，绝对无误地对所有对象的二进制秘密进行解码。这是有关波和辐射的语言，或者死者与活人的对话。这就是我们一起等待的地方，不管年龄，我们的车上装满了明亮色彩的货物。*
>
> ——Don Delillo，*White Noise*

仿制功能、匿名转信器、还有图片隐写术允许人们建立文件并把它们隐藏起来。如果没人能找到它们，就没有人需要知道密钥。这可能为罪犯提供了强有力的工具。

这儿有些安慰。使用电话窃听并不能停止街头突发暴力事件。劫匪、强奸犯、盗贼还有许多不依赖于通信犯罪的罪犯。对于每个人的一些绝大多数的重要问题和一些要求警察勤劳工作的事情，这些是可论证的。本书里没有什么工具可以影响到街道势力的平衡。

并不是禁止加密术和隐写术就可以阻止恐怖主义。劫持犯使用小刀和诡计，不是一组围绕在隐写空间底层的一组警句。笔并不是总比切纸刀强大。

事实上，很少数的犯罪是要通过搭线窃听解决的，因为只有很少量的犯罪依靠通信和信息交换。官员行贿，例如，只有两个人秘密地坐在一起并取得一致。如果不能通过钱来追查（这通常很难做到），那么只有记录谈话和某人作证的方法可以证明犯罪的发生。在某些州和某些州内的警察并不允许使用窃听技术。一些警官认为，这不是一个意外。这些州的政客意识到窃听的真正目标只能是政客。他们是议会中最先削弱法律的人。几乎所有别的犯法包括一些物理行为这样就会留下别的痕迹。

这个技术为创建秘密交易的能力是使人麻木的。惟一的安慰这些处理是在过去，只有傻子才会相信不是在将来。与会人们面对面进行违法商业交易，然当这有其他的利益。你可以亲自审判这些人。当然，这样交易的地方，例如酒吧，提供饮料和日常食品。你不能把它带进电脑空间。

事实上，骗子过去为了躲避窃听是很容易的，由全国步枪射击运动协会（NRA）引发了类似普遍意见。如果加密术被宣布为不合法，只有歹徒将拥有加密术。谋杀犯、走私犯或者小偷将可能不用对违法感到太犹豫，只需简单地管理他们发送位的来回而已。正直的人遵守任何法律保持加密术将发现他们自己很容易就能找到想偷窃他们机密的人。

当写本书的时候，我思考所有的这些问题。我几乎觉得争论的真正重点不是技术。如果罪犯可以常常逃避法律，他就并不真正需要用这些技术改变。警方常常被迫适应某项新技术，在这儿他们同样被迫这样做。

最后，我开始尝试怎样去平衡权力关系。如果权力可以阻止，那么滥用权力将会减少。如果个别人可以控制他们的事物，他们就会被别人减弱他们的优势。如果他们被迫公开工作，那么他们更像是被控制了。

不老实的人永远不会屈服于法律，告诉他们不要使用任何形式的隐写术。陈腐的歹徒声称10 000只香蕉的装船抵达将是我们永远的工作。问题就在于正直的人是否有权使用这些工具保护他们的秘密。加密术和隐写术给了个别人这些权力，好的，或者坏的，那将是这种权力最佳的去处。

跋

本书的一个评论家要求写一篇后记，并开玩笑说本书应该是“每个恐怖分子的床头必读。”然后他又说，“还有所有的自由战士、好莱坞主管、警方官员、虐待狂、首席信息指挥官和需要处处保密的任何人。”

谁知道你是不是一个恐怖分子，或许是个自由战士？但本书仅仅谈论技术，而技术是中性的。它教你如何改变拼写结构从而使数据看起来完全像别的东西。或许你对这些想法有一些好的计划，比如你要揭露本地化工厂随意排放有毒废弃物；或者你有一个充满了恶意的阴谋，你来不及图谋一个疯狂的计划；或许你是阴谋集团的一个执行者，正在用这些加密算法计划在何时何地排放有毒废弃物等，但技术只是中性的。

这是一些人的动力所在，他们情愿相信信息是有序的、结构化的、有组织的，全都是向上的、是正确的。我们想像电脑和它们有关这个世界琐事的海量存储将维持我们的安全，保护我们走向那些光明正大的目标，尽管我们可能不知道目标是什么。我们希望政府、银行、保险公司、零售商店、医生和几乎我们每一个人都能控制数据库，给我们一个完美的有序世界。

唉，没有什么可以远离真理。尽管位可以隐藏多重意义，它们可以认为是开或关、真或假、0或1，尽管位能够携带秘密信息和隐藏真相。信息并不像它看起来那样肯定和准确，有时雪茄携带大量的意义，但有时候雪茄就仅仅是雪茄，有时候甚至连雪茄也不是。

对于所有的这些，只有人才能对它作出判断，只有人才能够判断出“雪茄”是一种隐诲的暗示还是一种含有尼古丁的物体。我们希望人工智能和数据库引擎将能够分析所有的数据、所有的事实和所有的位，并确认恐怖分子必须受到严厉的打击，而好人必须受到帮助。信息分离端需要另一种特别的调节器。而你，本书的读者，是决定如何使用本书里的信息的人。你可以纠正错误行为，协调一场婚礼，计划一份永久的爱情，调整一个卑鄙的计划。技术只是中性的，本书只是纸上的方程式，而你，将决定这些方程式对这个世界的意义。

附录A Java仿真编码

在第7章我们描述了与上下文无关的基于语法的仿真函数，这里是这些仿真函数的一个Java版本的源代码。这个代码1997版权所有者是Peter Wayner，但是只要你遵守所有的美国出口法律，并且把所有的变化看成是你自己的，那么我们都欢迎你使用它。它是具有函数功能的，但是它使用了一个更好的随机数字生成器，一个检测词意连贯的语法机制以及一个更有效的分析。你可以找到许多改进它的方法。

BitInput Interface

```
public interface BitInput {
  public boolean nextBit();
  public void startBitSource(String s);
  public void startBitSource();
  public boolean atEnd();
    /// Returns true if everything has been outputted.
}
```

CTMimicCentral Class

```
import java.applet.*;
import java.io.*;
import java.net.*;
import java.awt.*;

public class CTMimicCentral extends Applet {
  TableSetter ts;
  OutSpitter os;
  MimicParser mp;

  Label mainAnnounce;
  TextField inputLine,outputLine;
TextArea outputMimic, inputMimic;
  // The main part of the input
Button inMimic,outMimic;
  // For getting to go.
boolean startedUp=false;

public void init(){
  this.setBackground(Color.white);
  ts=new TableSetter(getCodeBase());
  os=new OutSpitter();

  add(inputLine=new TextField("PUT MESSAGE HERE"));
  add(outputLine=new TextField(""));
  add(outputMimic=new TextArea(""));
  //add(inputMimic=new TextArea(""));

  add(inMimic=new Button ("Push for Mimicry"));
  add(outMimic=new Button("Remove Mimicry"));
```

```
  add(mainAnnounce=new Label("Loading Table. Please wait."));

  ts.setAnnounceHere(mainAnnounce);
  ts.run();

  startedUp=true;

}

public void layout(){
   int textWidth=400;
  if (startedUp){
    mainAnnounce.reshape(0,0,textWidth,20);
    if (inputLine!=null)
      {inputLine.reshape(5,20,textWidth,20);}
    if (inMimic!=null)
      {inMimic.reshape(5,45,200,18);}
    if (outputMimic!=null)
      {outputMimic.reshape(5,70,textWidth,150);}
    if (inputMimic!=null)
      {inputMimic.reshape(5,220,textWidth, 150);}
    if (outMimic!=null)
      {outMimic.reshape(5,240,200,18);}
    if (outputLine!=null)
      {outputLine.reshape(5,270,textWidth,20);}
  }
}
public void start(){

}
public boolean action(Event e, Object arg){
  boolean answer=false;
  if (ts.tableLoaded) {
    if (e.target == inMimic){
      os.setTableSource(ts);

        os.setTableSource(ts);
        os.setOutputTextArea(outputMimic);
        os.DoSomeMimicry(inputLine.getText());
        answer=true;

      } else if (e.target == outMimic){
        mp=new MimicParser(ts,outputMimic.getText());
        mp.setOutputLocation(outputLine);
        mp.DoItAllLoop();
        answer=true;
      }
    }
    return answer;
  }

}
```

Globals Class

```
import java.awt.*;
import java.applet.Applet;

 class Globals {
  public static final char StoppingCharacter = '/';
```

```
// In the definition of section where the grammar is defined,
// this character is used to separate the different
// productions that could occur. When it occurs twice,
// it signifies the end of the production...
 public static final char VariableSignifier = '*';
  // If the first character of a word equals this, then the program
  // shall treat the word as a variable which
  // will undergo more transformations. Note the program
  // ASSUMES that a variable is one word.
 public static final char EqualityCharacter = '=';
  // There is a character that signifies the equality
  // between a variable and set of productions.
 public static final char EndOfFileSignifier =  (char)(0);
  // This is passed back from NextWord when
  // it finds it is at the end of the file...
 public static final String NullWord = " ";
  // This is what comes back from NextWord
  // if it can't find something...
 public static final char Space = ' ';
  // This is something different, but the same.
  // It is the same thing, but used in a different fashion.

public static final boolean AddCarriageReturns=true;
  // If we want to do some word wrapping, this is inserted.
public static final int RightMargin=50;
  // The margin for wrapping.
}
```

WordNode Class

```
class WordNode {
 String w1;

   // The string being stored here.
 WordNode next;
   // What comes next in the list. null is nothing.

 public WordNode(){
   w1="";
   next=null;
 }
 public void setW1(String s){
   w1=s;
 }
 public WordNode (String s){
   w1=s;
   next=null;
 }
 public WordNode (String s, WordNode n){
   w1=s;
   next=n;
 }
 public void setNext(WordNode n){
   next=n;
 }
 public WordNode getNext(){
   return next;
 }
}
```

BitNode Class

```
class BitNode {
 int bitNumber=-1;
   // This is the number of a central register of bits.
 boolean polarity=true;
 double probability;
   // The total probability of all subordinate nodes.
 BitNode up;
   // What comes up.
 BitNode left,right;
   // For building a tree.
 ProductionNode theProductionNode;
     /*Well, technically, I could get away
       with storing this pointer in either left or
      right because left and right will equal nil if
       and only if TheProductionNode <> nil, but
      I don't feel like packing this too tightly
       right now. I'm being lavish with memory.*/

 public BitNode(){
   up=null;
   left=null;
   right=null;
   bitNumber=0;
   polarity=true;
   probability=1.0;
 }
 public void setProbability(double d){
   probability=d;
 }
}
```

ProductionNode Class

```
class ProductionNode {
double probability;
  // The Probability that this particular
  // production will be chosen.
BitNode itsBit;
  // Follow this up to the root to find the bits
  // associated with this production.
WordNode theWords;
  // This is the list of words that come from
  // the Production. Variables are at the end.
ProductionNode next;
  // This is the next in the list.


public ProductionNode(){
  probability=1.0;
  itsBit=null;
  theWords=null;
  next=null;
}
public void setProbability(double d){
  probability=d;
}
public void setNext(ProductionNode n){
```

```
    next =n;
  }
  public ProductionNode getNext(){
    return next;
  }
  public void setTheWords(WordNode w){
    theWords=w;
  }
}
```

VariableNode Class

```
class VariableNode {
  String wl;
    // This is the identity of the variable.
  ProductionNode productions;
   // This is the list of productions associated
   // with the variable.
 BitNode itsBitRoot;
   // This is the top of the bit tree. When
   // random characters are being generated,
   // it follows this down to the bottom.
 VariableNode next;

 public VariableNode(){
   wl="";
   productions=null;
   itsBitRoot=null;
   next=null;
 }

 public void setProductions(ProductionNode p){
   productions=p;
 }

 public ProductionNode getProductions(){
   return productions;
 }

 public void setWl(String w){
   wl=w;
 }

 public void setNext(VariableNode n){
   next=n;
 }

 public VariableNode getNext(){
   return next;
 }
}
```

MimicProdNode Class

```
class MimicProdNode{
  WordNode ww;
  MimicProdNode next;
```

```
  public  MimicProdNode() {
    ww=null;
    next=null;
  }

  public void setNext(MimicProdNode n){
    next =n;
  }

  public MimicProdNode getNext(){
    return next;
  }

  public void setWW(WordNode a){
    ww=a;
  }

  public WordNode getWW(){
    return ww;
  }
}
```

MimicParseFrame Class

```
class MimicParseFrame {
  VariableNode theVariable;
  WordNode theWordsToMatch;
    // Once a production is matched,
    // this baby is loaded with words to check.

  public MimicParseFrame(){
    theVariable=null;
    theWordsToMatch=null;
  }
}
```

NextWordResponse Class

```
class NextWordResponse {
  String w;
  char stopChar;
  boolean doubleStop;

  public NextWordResponse(String s, char stop, boolean ds){
    w=s;
    stopChar=stop;
    doubleStop=ds;
      // set to be true when we hit a double slash.
  }
}
```

CTTableSetter Class

```
import java.applet.*;
import java.io.*;
import java.net.*;

public class CTTableSetter {
  WordEater we;
  NextWordResponse n;

  public TableSetter(){
    we= new WordEater();
    we.initializeScanHashTable();
  }
}
```

MyIntegerWrapper Class

```
import java.io.*;
import java.net.*;
import java.awt.*;
class MyIntegerWrapper {
  int x;
    // Used to pass by reference.

  public MyIntegerWrapper(int zz){
    x=zz;
  }

  public void setX(int y){
    x=y;
  }
  public int getX(){
    return x;
  }
}
```

MimicParser Class

```
public class MimicParser {

  public final static int spacer=1;
  public final static int normal=2;
  public final static int stopper=3;

  public final static char equalityCharacter='=';
  public final static char stoppingCharacter='/';

  public final static int MaxLookAhead=200;
  public final static int StandardLookAhead=20;
      /* MaxLookAhead is the absolute maximum permitted
      by the sizes of the array. StandardLookAhead is
      the usual amount to search. Note that the running
      time is potentially exponential in the
      LookAhead. Don't be greedy. Sorry I didn't write
      a better algorithm. */

    public RandomBits rb=new RandomBits();
      // For synchronizing things. ...
    public boolean debug=false;
    public int debugStackDepth=0;
      // For controlling the printout.

    public int offsetTraffic=0;
      // Used for passing information.

  public TableSetter ts;
    // Where the information is kept.

  public TextField outputLocation=null;
    // Where the data goes.

  public String[] LookAheadTable;
        /* The next MaxLookAhead words are kept in this
        circular array. When the end of file is found, the
        table contains nullWord. */
```

```
public int LookAheadOffset=0;
   /* Keeps track of where the first word will be. */

public boolean FoundAmbiguity=false;
      /* When the machine starts to do some parsing and
      discovers that there are two different paths for
      each production, this baby is set to true. */

public int SoftMaxLookAhead;
      /* This allows the parser to set its lookahead on
      the fly if it wants to increase it. */

public boolean ReachedEndOfFile=false;
  /* Set true when everything is exhausted... */

StringBuffer outputStrings;
  /// Where the data is going.

int tempOutputBits=0;
int tempOutputBitsPosition=0;

public void openOutputFile(){
  outputStrings= new StringBuffer("");
}

String toBeParsed;
  // What will be parsed.
int toBeParsedPosition=0;
  // The current characters.

public int[] scanTable=new int[256];
  // This is a scan table used for parsing.

public void setOutputLocation(TextField f){
  outputLocation=f;
}


public void initializeScanHashTable()\{
    /* The Scanning Hash Table has 256 entries. They
    identify a character as either a letter, stop
    character,} a space or a comment. */
  int i;
  for (i=14; i<=255; i++){
    scanTable[i]=normal;
  }
  for (i=0; i<=13; i++){
    scanTable[i]=spacer;
  }
     scanTable[(byte)(equalityCharacter)] = stopper;
     scanTable[(byte)(stoppingCharacter)]  = stopper;
     scanTable[(byte)(' ')] = spacer;
}
public void debugFirstPrint(String s){
  for (int i=0; i<debugStackDepth;i++){
    System.out.print("-");
  }
  System.out.print(s);
}
```

```
  public void debugPrintln(String s) {
    for (int i=0; i<debugStackDepth;i++){
      System.out.print("-");
    }
    System.out.println(s);
  }

  public String parserNextWord(){
    // Pull off the next word.
    String answer ="";
    while ((toBeParsedPosition < toBeParsed.length())
       && (scanTable[toBeParsed.charAt(toBeParsedPosition)] != normal)) {
      toBeParsedPosition++;
    }
    while ((toBeParsedPosition < toBeParsed.length())
       && (scanTable[toBeParsed.charAt(toBeParsedPosition)] == normal)) {
      answer=answer+toBeParsed.charAt(toBeParsedPosition++);
    }
    if (toBeParsedPosition >= toBeParsed.length()){
      ReachedEndOfFile = true;
    }
    return answer+" ";
  }


/* Sets up the lookahead buffer
to keep all of the words in place. */

  public void InitLookAhead(){
    int i;
    char Stopper; /* to be ignored. */
    LookAheadTable = new String[MaxLookAhead+1];

    LookAheadOffset = 0;
    for (i = 0; i<= MaxLookAhead - 1; i ++){
        LookAheadTable[i]= parserNextWord();
      }
  }

   public MimicParser(TableSetter tt, String doMe){
      ts=tt;
      toBeParsed=doMe;
      toBeParsedPosition=0;
      initializeScanHashTable();
      InitLookAhead();
     }
 public String  TimeDelayNextWord(){
   String result;

   result=LookAheadTable[LookAheadOffset];
   LookAheadTable[LookAheadOffset]=parserNextWord();

   LookAheadOffset = (LookAheadOffset + 1) % MaxLookAhead;
   return result;
 }


   public char convertNumToChar(int c){
     // Just converts things back.
```

```
switch(c){
  case 23:
  case 31:
  case 26:
  case 0: return ' ';
  // 29-11101-10111-23
  // 31-11111-11111-31
  // 11-01011-11010-26
  // 00000-00000
  case 16: return 'A';
  // 00001-10000-16
  case 8: return 'B';
  // 00010-01000-8
  case 24: return 'C';
  // 00011-11000-24
  case 4: return 'D';
  // 00100-00100-4
  case 20: return 'E';
  // 00101-10100-20
  case 12: return 'F';
  // 00110-01100-12
  case 28: return 'G';
  // 00111-11100-28
  case 2: return 'H';
  // 01000-00010-2
  case 18: return 'I';
  // 01001-10010-18
  case 10: return 'J';
  // 01010-01010-10
  case 15: return 'K';
  // 01011-11010-26
  // K is now 30-11110-01111-15
  case 6: return 'L';
  // 01100-00110-6
  case 22: return 'M';
  // 01101-10110-22
  case 14: return 'N';
  // 01110-01110-14
  case 30: return 'O';
  // 01111-11110-30
  case 1: return 'P';
  // 10000-00001-1
  case 17: return 'Q';
  // 10001-10001-17
  case 9: return 'R';
  // 10010-01001-9
  case 25: return 'S';
  // 10011-11001-25
  case 5: return 'T';
  // 10100-00101-5
  case 21: return 'U';
  // 10101-10101-21
  case 13: return 'V';
  // 10110-01101-13
  case 29: return 'W';
  // 10111-11101-29
  case 3: return 'X';
  // 11000-00011-3
  case 19: return 'Y';
  // 11001-10011-19
  case 11: return 'Z';
```

```
    // 11010=01011=11
    case 27: return '-';
    // 11011=11011=27
    default: return '*';
  }
}


public void storeOutputBit(boolean b){
  if (b) {
    tempOutputBits=(tempOutputBits << 1) +1;
  } else
    tempOutputBits = (tempOutputBits <<1);
  tempOutputBitsPosition++;
  if (tempOutputBitsPosition>=5){
    tempOutputBitsPosition = 0;
    outputStrings.append(convertNumToChar(tempOutputBits));
    tempOutputBits = 0;
  }
}

public boolean RandomBit(int a){
  return true;
  // No random synchronization at this time.
}


public void ProductionToBits(ProductionNode p){
  BitNode BitPointer;
    /* This will lead the way. */
  long TempBit;
    /* This will hold the bits. */
    int BitCounter=0;
      /* This is position of the next bit to be stored. */
    boolean[] store=new boolean[30];
      // This should be enough.
    rb.updateRandomBits();
      /* Get a new group of random
         bits for the random generator. */
    TempBit=0;
    BitCounter=0;
    BitPointer=p.itsBit;
      /* We're not concerned about the first node because
           it is just an interface. */
    //debugFirstPrint("Bits:");
    while (BitPointer.up != null) {
        if (BitPointer.polarity) {
            store[BitCounter++]=(rb.randomBit(BitPointer.up.bitNumber) && true);
            // System.out.print("T");
          } else {
            store[BitCounter++]=(rb.randomBit(BitPointer.up.bitNumber) ^ true);
            // System.out.print("F");
          }
        BitPointer= BitPointer.up;
    }
    for (int i=BitCounter-1;i>=0;i--){
      storeOutputBit(store[i]);}
      // These are backward. We can reverse them here.
  //  System.out.print(" ");
  }
```

```
      /* Imagine this case. You are now trying to decide
      whether the next token, say "Ernest", came for the
      production of a variable "*Dudes" or "*Duds". It
      could be from either. This tries to lookahead to
      see if there is any clue that says, Hey, it can't
      be "*Dudes" because the production of the token
      "Ernest" is always followed by the token "Rex" to
      indicate his stature. */

 public ProductionNode TokenInVariable(MyIntegerWrapper MoreOffsetWrapper,
VariableNode v){
  ProductionNode ProductionNumber;
      /* Well not as much a number as a pun. */
  boolean OneFound;
      /* This is just set to look for
      ambiguities...Problems, you know.... */
      int  RealOffset;
      /* This holds the temporary Offset that is passed
      to the CheckWordList function. */
    MyIntegerWrapper tempOffset=new MyIntegerWrapper(0);
  ProductionNode result = null;


  ProductionNumber = v.productions;
  OneFound = false;
  RealOffset = MoreOffsetWrapper.getX();

  while ((ProductionNumber != null) && !OneFound) {
      tempOffset.setX(RealOffset);
      debugStackDepth += 2;
      if (CheckWordList(tempOffset, ProductionNumber.theWords)) {
    if (debug) {
      WordNode temp = ProductionNumber.theWords;

        debugPrintln("TokenInVariable found production of "+v.wl+":" + tempOffset+":");
        while (temp != null) {
             System.out.print(temp.wl);
             temp = temp.next;
          }
         System.out.println(" ");
      }

        OneFound = true;
        MoreOffsetWrapper.setX(tempOffset.getX()-1);
        result = ProductionNumber;
        }
              debugStackDepth -= 2;
      ProductionNumber = ProductionNumber.next;
   }
return result;
}

      /* This compares the words in the word list with
      the words in the lookahead buffer.  If they all
      match, then BINGO! */

public  boolean CheckWordList(MyIntegerWrapper offset, WordNode www) {
  WordNode checkMe = www;
  int newOffset = offset.x;
  MyIntegerWrapper passItOn=new MyIntegerWrapper(0);
  // debugFirstPrint("Offset:"+passItOn.x+" Comparing word:"+checkMe.wl);
```

```
    while ((newOffset < SoftMaxLookAhead )&& (checkMe != null)) {
        if (checkMe.wl.charAt(0) == Globals.VariableSignifier) {
          passItOn.setX(newOffset);
                debugStackDepth += 2;
          if (TokenInVariable(passItOn, ts.FindVariable(checkMe.wl)) == null) {
              offset.setX(passItOn.getX());
              return false;
           }
                debugStackDepth -= 2;
                      offset.setX(passItOn.getX());

          newOffset= passItOn.getX()+1;
          checkMe = checkMe.next;
        } else if (LookAheadTable[(newOffset + LookAheadOffset)
              % MaxLookAhead].compareTo(checkMe.wl) != 0) {
     //    System.out.println(""+checkMe.wl+"::FAILED");
        offset.setX( newOffset);
          return false;
        } else {
        //  System.out.print(""+checkMe.wl);
          newOffset ++;
          checkMe = checkMe.next;
        }
    }


    offset.setX(newOffset);
    // debugPrintln("Found at offset:"+newOffset);
  return true;
  }

        /* The Serious Version Checks for Parsing
        Ambiguities.... This is only done on the first
        call. You could get rid of this and just let the
        program choose the first production it finds. This
        will be faster, but it might lead to errors. I've
        chosen to include this because it is
        computationally difficult (at least for now) to
        ensure that CFLs are non-ambiguous. */

   public ProductionNode SeriousTokenInVariable(int MoreOffset, VariableNode v){
    ProductionNode ProductionNumber; /* Well, not as much a number as a pun. */
    boolean OneFound;
         /* This is just set to look for ambiguities. Problems, you know.... */
    int  RealOffset, bestOffset;
    MyIntegerWrapper tempOffset=new MyIntegerWrapper(0);
        /* This holds the temporary Offset that is passed to the CheckWordList
            function. */
    ProductionNode result = null;

    ProductionNumber = v.productions;
    OneFound = false;
    RealOffset = MoreOffset;
        bestOffset = MoreOffset+1;
        tempOffset.setX(RealOffset);

    while (ProductionNumber != null) {
      tempOffset.setX(RealOffset);
        if (OneFound) {
        if (CheckWordList(tempOffset, ProductionNumber.theWords)) {
```

```
        if (tempOffset.getX() == bestOffset) {
        System.out.println("Parsing Ambiguity Here!!! Try growing the lookahead. ");
        FoundAmbiguity = true;
        return null;
      } else if (tempOffset.getX() > bestOffset) {
        bestOffset = tempOffset.x;
        result = ProductionNumber;
      }
          /* else ignore shorter match */
        }
      }  else if (CheckWordList(tempOffset, ProductionNumber.theWords)) {
        OneFound = true;
        bestOffset = tempOffset.x;
        result = ProductionNumber;
      }
      ProductionNumber = ProductionNumber.next;
    }
    if (debug) {
    WordNode temp = result.theWords;
      //System.out.print("TokenInVariable found production of "+v.wl+":" +
tempOffset+":");
      while (temp != null) {
           System.out.print(temp.wl);
          temp = temp.next;
        }
     // System.out.println(" ");
    }
return result;
}


/* This routine takes the frame and figures out that pointer would match... */
 public void DoFrame(MimicParseFrame f){
  ProductionNode Prod;  /* This is the location of the pointer... */
  MimicParseFrame NewFrame; /* If a new frame is needed... */
  int CurrentDepth; /* Used to halt the spread of the search. */
  MyIntegerWrapper curDepth=new MyIntegerWrapper(0);

  FoundAmbiguity = false;
  SoftMaxLookAhead = StandardLookAhead;
  CurrentDepth = 0;
  // debugPrintln("Looking up variable:"+f.theVariable.wl);
  debugStackDepth+=3;
  curDepth.setX(CurrentDepth);
  Prod = TokenInVariable(curDepth, f.theVariable);

        /* I realize that it is a bit of an overkill to
        use a big hammer like SeriousTokenInVariable, but
        it is late at night and I don't want to bother
        writing an elegant method that uses breadth first
        instead of depth-first. */

    debugStackDepth-=3;
  if (FoundAmbiguity) {
     FoundAmbiguity = false;
      SoftMaxLookAhead = MaxLookAhead;
      /* This just doubles the lookahead for the fun of it... */
      CurrentDepth = 0;
        curDepth.setX(CurrentDepth);
      Prod = TokenInVariable(curDepth, f.theVariable);
```

```
    }
    if (Prod == null) {
      System.out.println("Can't seem to find a production for:"+f.theVariable.wl);
       curDepth.setX(CurrentDepth);
       Prod = TokenInVariable(curDepth, f.theVariable);
      //longjmp(LABEL\_199, true);
     } else if (!FoundAmbiguity) {

      /* We've found something.... */
     ProductionToBits(Prod);
      /* Store the bits... */
       NewFrame= new MimicParseFrame();
     f.theWordsToMatch = Prod.theWords;
     while (f.theWordsToMatch != null) {
       if (f.theWordsToMatch.wl.charAt(0) == Globals.VariableSignifier) {
         NewFrame.theVariable = ts.FindVariable(f.theWordsToMatch.wl);
              debugStackDepth+=3;
             DoFrame(NewFrame);
             debugStackDepth-=3;

       } else {
         String theOther = TimeDelayNextWord();

           if (f.theWordsToMatch.wl.compareTo(theOther) != 0) {
       System.out.println("Problem in parsing the file. The Word "+f.theWordsToMatch
.wl+" doesn't belong here.");
       System.out.println("Supposed to match:"+theOther);
       //        longjmp(LABEL\_199, true);
         }
       }
       f.theWordsToMatch = f.theWordsToMatch.next;
      }
  }
}

public String stripSpaces(String  s){
  int i=s.length()-1;
  while (s.charAt(i)==' ') {
    i--;
  }
  return s.substring(0,i+1);
}

public void DoItAllLoop(){
  MimicParseFrame BaseFrame;
    /* This is the first frame allocated. */
  int i=0;
  rb.syncRandomBits();
  openOutputFile();
 BaseFrame = new MimicParseFrame();
  BaseFrame.theVariable = ts.GetStartVariable();
  do {
    DoFrame(BaseFrame);
    i++; // This is a safety net to pull us out of the loop.
    } while ((i<10) && !
        (ReachedEndOfFile &&
          LookAheadTable[LookAheadOffset].compareTo(Globals.NullWord) == 0));
    if (outputLocation!=null){
      outputLocation.setText(stripSpaces(outputStrings.toString()));
    }
  //  System.out.println("Finished.");
```

```
 // rb.outputDebugArray();
}

}
```

OutSpitter Class

```
import java.io.*;
import java.net.*;
import CTMimicGlobals;
import java.awt.*;
public class OutSpitter extends Thread {
  BitInput bitSource;
    // This gets the bits.
  TableSetter tableSource;
    // What we use to get everything in shape.
  StringBuffer outputForMimicry;
    // Where the data goes.

  TextArea outputTextArea=null;

  RandomBits rb;
    // Where the random bits come from
  MimicProdNode TheOutputStack;
     /* This is the stack of variables that are unproduced... */
   int CarriagePosition=0;
      /* This is the location of the last character printed
         out on the page. For justification. */

  int StackCount=0;


  public OutSpitter(){
    super();
    rb=new RandomBits();
  }

  public void setTableSource(TableSetter t){
    tableSource = t;
  }

  public void setOutputTextArea(TextArea t){
    outputTextArea = t;
  }



public void OpenForOutput(){
  outputForMimicry= new StringBuffer("");
}

public boolean RandomBit(int i){
  return true;
  // This is just a placeholder for a synchronized random number generator.
}

         /* This baby takes a variable and follows its way
         down to the production using the bit tree. */
```

```
public ProductionNode VariableToProduction(VariableNode v){
 BitNode Bitto; /* This is the position that the bit tree should follow. */

 rb.updateRandomBits(); /* Cycle the random number generator. */
 Bitto = v.itsBitRoot;
 while (Bitto.theProductionNode == null) {
     if (bitSource.nextBit()) {
        if (rb.randomBit(Bitto.bitNumber))
          Bitto = Bitto.right;
       else
           Bitto = Bitto.left;
     }  else {
       if (rb.randomBit(Bitto.bitNumber))
          Bitto = Bitto.left;
       else
          Bitto = Bitto.right;
     }
   }
 //  System.out.print(" ");
  return Bitto.theProductionNode; /* TheAnswer. */
}

/* This will do the correct thing with the word
node w. If it is a terminal, it will write it out.
Otherwise it will start a new frame... */

public  void DoWord(WordNode w){
  MimicProdNode StackFrame; /* For creating new ones... */

if (w.wl.charAt(0) == Globals.VariableSignifier) {
  StackFrame = new MimicProdNode();
  StackFrame.setNext(TheOutputStack);
  TheOutputStack = StackFrame;

StackFrame.setWW(VariableToProduction(tableSource.FindVariable(w.wl)).theWords);
  if (StackCount++ <500) {
    //System.out.println("Going down:"+w.wl);
   DoStack(StackFrame);
  //  System.out.println("Coming up:"+w.wl);
  } else {
  //  System.out.println("Ignoring:"+w.wl);
  }
  } else {
  /* Write it out. */
  outputForMimicry.append(w.wl);
  // System.out.print();
  /* Assuming there is a space at the end of each word. */
  if (Globals.AddCarriageReturns) {
    CarriagePosition += (w.wl.length());
    if (CarriagePosition > Globals.RightMargin) {
      // Not Sure How to do this...  fprintf(OpenMimicryFile, "\\n");

   //  System.out.println("");
      CarriagePosition = 0;
      }
    }
  }
}
```

```
public void DoStack(MimicProdNode s){
WordNode wurds;
 /* This just goes through the list of words on the stack until they are gone. */
wurds = s.getWW();
while (wurds != null) {
  //System.out.println("About to DoStack on word:"+wurds.w1);
  DoWord(wurds);
  wurds = wurds.getNext();
  }
 TheOutputStack = TheOutputStack.getNext();
}

public void DoSomeMimicry(String s){
  // This starts it up.
  rb.syncRandomBits(); // We're not going to do that right now.
  OpenForOutput();
  bitSource= new StringBufferBitInput();
  bitSource.startBitSource(s);
  //System.out.println("Initialized bit source. Read to start generating.----");

  while (!bitSource.atEnd()) {
    TheOutputStack = new MimicProdNode();

TheOutputStack.setWW(VariableToProduction(tableSource.GetStartVariable()).theWords);
      DoStack(TheOutputStack);
    }
  outputTextArea.setText(outputForMimicry.toString());
  //System.out.println();
  //rb.outputDebugArray();
  //  System.out.println();

}}
```

RandomBits Class

```
public class RandomBits {
     /*  This uses an outdated random number generator
     created by Stephen Wolfram. The original one was
     considered to be cryptographically secure, but
     others have found a way to break it. I wish I
     could use a better one, but I don't have time to
     research it. */

  int initialKey, randomBits;
    // For holding the information.

public boolean[] debugArray=new boolean[10000];
public int debugArrayPos=0;

public RandomBits(){
  initialKey=0xbaad111f;
  randomBits=initialKey;
}

public void syncRandomBits(){
  randomBits=initialKey;
  debugArrayPos=0;
}

public void updateRandomBits(){
  randomBits = (randomBits << 1) ^ ( randomBits | (randomBits >> 1));
}
```

```
  public boolean randomBit(int i){
    return true;
    /*  i = i % 32;

    if ((1 & (randomBits >> i)) == 1 ){
      debugArray[debugArrayPos++]=true;
      return true;
    } else {
      debugArray[debugArrayPos++]=false;
      return false;
    }
    */
  }

  public void outputDebugArray(){
    for (int i=0;i<debugArrayPos;i++){
      if (debugArray[i]){
        System.out.print('T');
      } else {
        System.out.print('f');
      }
    }
    System.out.println(' ');
  }
}
```

StringBufferBitInput Class

```
public class StringBufferBitInput implements BitInput {

  StringBuffer text;
    // This is what comes next.
  int nextBitsSource;
    // This is what is going out next.
  int nextBitsPosition;
    // This points to the bit to leave.

  boolean atTheEndOfBuffer=false;

     // Set to be true.
   int textPosition=0;
     // Where it is found.

   public boolean atEnd(){
     return atTheEndOfBuffer;
   }

   public void setText(StringBuffer s){
     text = s;
   }

   public void loadNextBits(char c){
     // This just inserts the right set of bits into place.
     // This version uses an abbreviated set of values between 0 and 31 to use only
 5 bits.
     // This is a short way of compressing things.

     nextBitsPosition=0;
     if (c==' ') {
       double r=Math.random();
       if (r<.25){
```

```
      nextBitsSource=0;
    } else if (r<.5) {
      nextBitsSource=29;
    } else if (r<.75) {
      nextBitsSource=11;
    } else {
      nextBitsSource=31;
  }} else {
  switch(c){
    case ' ': nextBitsSource=0; break;
    case 'A': case 'a': nextBitsSource=1; break;
    case 'B': case 'b': nextBitsSource=2; break;
    case 'C': case 'c': nextBitsSource=3; break;
    case 'D': case 'd': nextBitsSource=4; break;
    case 'E': case 'e': nextBitsSource=5; break;
    case 'F': case 'f': nextBitsSource=6; break;
    case 'G': case 'g': nextBitsSource=7; break;
    case 'H': case 'h': nextBitsSource=8; break;
    case 'I': case 'i': nextBitsSource=9; break;
    case 'J': case 'j': nextBitsSource=10; break;
    case 'K': case 'k': nextBitsSource=30; break;
    case 'L': case 'l': nextBitsSource=12; break;
    case 'M': case 'm': nextBitsSource=13; break;
    case 'N': case 'n': nextBitsSource=14; break;
    case 'O': case 'o': nextBitsSource=15; break;
    case 'P': case 'p': nextBitsSource=16; break;
    case 'Q': case 'q': nextBitsSource=17; break;
    case 'R': case 'r': nextBitsSource=18; break;
    case 'S': case 's': nextBitsSource=19; break;
    case 'T': case 't': nextBitsSource=20; break;
    case 'U': case 'u': nextBitsSource=21; break;
    case 'V': case 'v': nextBitsSource=22; break;
    case 'W': case 'w': nextBitsSource=23; break;
    case 'X': case 'x': nextBitsSource=24; break;
    case 'Y': case 'y': nextBitsSource=25; break;
    case 'Z': case 'z': nextBitsSource=26; break;
    case '-':nextBitsSource=27;
    default: nextBitsSource=28;
  }
 }
}


public boolean nextBit(){
  boolean answer;
  if ((nextBitsSource & 1)!=0){
    answer=true;
  } else {
    answer = false;
  }
  nextBitsPosition++;
  if (nextBitsPosition>=5) {
    textPosition++;

    if (textPosition>=text.length()){
      atTheEndOfBuffer = true;
      loadNextBits(' ');
    } else {
      loadNextBits(text.charAt(textPosition));
    }
```

```
    } else {
      nextBitsSource = nextBitsSource>>1;
    }
  /*  if (answer) {
      System.out.print("T");
    } else {
      System.out.print("F");
    } */
    return answer;
  }

  public void startBitSource(){
    textPosition=0;
    text = new StringBuffer("hi mom.");
    loadNextBits(text.charAt(textPosition));
  }
  public void startBitSource(String s){
    textPosition=0;
    text = new StringBuffer(s);
    loadNextBits(text.charAt(textPosition));
  }
}
```

TableSetter Class

```
import java.io.*;
import java.net.*;
import java.awt.*;

public class TableSetter extends Thread{
  boolean tableLoaded=false;
  WordEater we;
  NextWordResponse n;
  VariableNode CurrentVariable;
    // What is currently being parsed.
  VariableNode VariableListRoot=null;
      /// The list of all variables.
  boolean WasThereNoError=true;

  URL base;
    // This must be passed down because
    // only applets can discover their URL base.

  Label announceHere=null;

  public void setAnnounceHere(Label s){
    announceHere=s;
  }

      /* This returns the variable that starts out every
      production. This is just set to be the first one
      alphabetically in the list. It would be possible to
      put some sort of random selection here too if you
      wanted to add an additional signifier that said "I'm
      a good candidate to start a production." */

 VariableNode GetStartVariable()  {
   return VariableListRoot;
 }
```

```
public boolean loadTable(){
  //This function tries to load the information
  // from the currently opened file into table.
  // Returns true if it succeeds. False if it signals an error
     boolean looping;

     VariableListRoot = null;
     looping = true;
     while (looping) {
        try {
          sleep(40);
        } catch (InterruptedException e) {
        }
       if (HandleFirst()) {
         looping = HandleProduction();
       } else {
          looping = false;
       }
      }
     BuildBitTable();
      // PrintVariableList(VariableListRoot);
      return  WasThereNoError;
    }

 public void run(){
   //System.out.println("Starting the run thing.");
   if (announceHere!=null){
     announceHere.setText("Loading Grammar Table. Please Wait.");
   }
     try {
           sleep(40);
         } catch (InterruptedException e) {
         }
   we.initializeScanHashTable();
   we.openGrammarURL("BASE.CFL",base);
       //System.out.println("Opened the grammar thing.");

   loadTable();

   if (announceHere!=null){
     announceHere.setText("Table Loaded. Ready to go.");
   }
   tableLoaded=true;
 }

 public TableSetter(URL u){
   base=u;
   we= new WordEater();

 }

 public void AddVariable (VariableNode v ){
   //This adds it to the root.}
      VariableNode previous, node;
   //For scanning along the list.}
      int Relativity;

     node = VariableListRoot;
     previous = null; // Must init it.
     if (VariableListRoot == null) {
```

```
      VariableListRoot = v;
      v.setNext(null);
    } else if (node.wl.compareTo(v.wl)>0) {
      VariableListRoot = v;
        v.next = node;
    } else {
      while (node != null ) {
       Relativity = node.wl.compareTo(v.wl);
           if (Relativity<0) {
            previous = node;
            node = node.next;
           } else  if  (Relativity == 0) {
            System.out.println("'"+  v.wl +"'  has been previously defined.");
         } else {
           previous.setNext(v);

                v.setNext(node);
              node = null;
            v=null;
          }
          }
          if (v != null) {
          previous.setNext(v);
          v.setNext(null);
          }
      }
    }

      /* Looks up the list and finds the variable
      corresponding to it. Note that everything is
      extremely slow to just keep this in a list! */

 VariableNode FindVariable(String name){
  VariableNode temp;
    /* Sort of a stunt double for FindVariable. */
  int relativity;
    /* Just for storing the relative differences between strings. */

  temp = VariableListRoot; /* Start at the very beginning. */
  while (temp != null) {
        relativity = name.compareTo(temp.wl);
      if (relativity > 0) {
        temp = temp.next;
      } else {
        if (relativity == 0)
           return temp;
        else
          return null;
        }
      }
    return null;
  }

  public boolean HandleFirst(){
   // The first word in a line must be a variable.
   // This will be used to set up the variable list.

     VariableNode v;
    // This is what is going to get built.}
     NextWordResponse wa,wb;
```

```
boolean answer=true;

  wa = we.nextWord();
  if (wa.stopChar == Globals.EndOfFileSignifier) {
    if (! wa.w.equals(Globals.NullWord)) {
     System.out.println("Unexpected end of the file.");
}

    return false;
    }
    if ((wa.w.equals(Globals.NullWord)) ||
       (wa.w.charAt(0) != Globals.VariableSignifier)) {
     System.out.println("Here's the first char:"+wa.w.charAt(0)+
                        "::"+ Globals.VariableSignifier+"::");
     System.out.println("Expected a variable name at the beginning of the line:");
    } else {
       v=new VariableNode();
      v.setW1(wa.w);
    // Now add it to the list in the right place.}
      AddVariable(v);
      CurrentVariable = v;
    }
    if (wa.stopChar != Globals.EqualityCharacter) {
      do {
          wa = we.nextWord();
      } while ((wa.stopChar == Globals.Space) && (wa.w.equals(Globals.NullWord)));
       if (! wa.w.equals(Globals.NullWord)) {
         System.out.println("The Variable should only be one word. "+ wa.w+" is too
much. Error in line:");
       }
     }
     return true;
    }

    public void  SkipToEnd(){
    // This procedure just keeps hitting NextWord until
    // it hits the double StoppingCharacter, which when
    // I wrote this line was defined to be '//'.
    NextWordResponse wa;
    char previous;
    boolean dontStopNow=true;

    // WasThereNoError := false;

     wa = we.nextWord();
     do {
       previous = wa.stopChar;
       wa = we.nextWord();
       if (wa.stopChar==Globals.EndOfFileSignifier) {
          dontStopNow=false;
       } if ((wa.w.equals(Globals.NullWord))
               && (previous == Globals.StoppingCharacter)
               && (wa.stopChar== Globals.StoppingCharacter)) {
        dontStopNow=false;
       }
     } while (dontStopNow);
    }

  /*public double wordToValue(String s){
   double answer = Double.valueOf("0"+s).doubleValue();
   return answer;
```

```
  }
  */

 public double wordToValue(String s){
   double answer = 0.0;
   boolean fraction=false;
   int count = 0;
   for (int i=0;i<s.length();i++){
     switch (s.charAt(i)){
       case '0':
         answer=10*answer+0;break;
       case '1':
         answer=10*answer+1;break;
       case '2':
         answer=10*answer+2;break;
       case '3':
         answer=10*answer+3;break;
       case '4':
         answer=10*answer+4;break;
       case '5':
         answer=10*answer+5;break;
       case '6':
         answer=10*answer+6;break;
       case '7':
         answer=10*answer+7;break;
       case '8':
         answer=10*answer+8;break;
       case '9':
         answer=10*answer+9;break;
       case '.':
         fraction = true;break;
     }
     if (fraction) count++;
   }
   for (int i=0;i<count;i++){
     answer=answer/10;
   }
   return answer;
 }

public boolean HandleProduction(){
 // Keeps Adding Production until it
 // encounters a double Stopping Character.

  WordNode   LastAddedWord=null;
   // This is just a place keeper which points
   // to the last word added so the next can
   // be updated when another one is added.
  ProductionNode TheProduction=null;
   //This is where the info goes.
  NextWordResponse wa;
   //For local reasons.

  boolean startedVariables=false;
   //Variables can only come at the end of productions.
  boolean IsError;
   //If there is a problem this get's set to be true.
  boolean answer=true;
```

```
   do {
      do {
        wa = we.nextWord();
        if (wa.stopChar == Globals.EndOfFileSignifier) {
           if (LastAddedWord != null) {
             System.out.println("Just parsed something left incomplete by the end of
the file:");
             // PrintWordList(LastAddedWord);
             System.out.println("Unexpected end of the file.");
            answer = false;
           }
           return answer;
      }
        if (! wa.w.equals(Globals.NullWord)) {
           if (LastAddedWord == null ) {
           // Start a new production.
             if (wa.w.charAt(0) == Globals.VariableSignifier){
               System.out.println("'The first word of a production, '"+ wa.w+"' cannot
be a variable. Ignoring Production.");
             SkipToEnd();
               wa.stopChar = Globals.StoppingCharacter;
               wa.w = Globals.NullWord;
         } else {
            // System.out.print(":"+wa.w);
             TheProduction = new ProductionNode();
                 TheProduction.setNext(CurrentVariable.getProductions());
                 CurrentVariable.setProductions(TheProduction);
                 // Put it at the beginning of the list...
             LastAddedWord = new WordNode(wa.w);
             TheProduction.setTheWords(LastAddedWord);
             startedVariables = false;
         }
        }  else {
            LastAddedWord.setNext(new WordNode(wa.w));
                LastAddedWord = LastAddedWord.getNext();
                if (wa.w.charAt(0) == Globals.VariableSignifier) {
                   if  (! startedVariables) {
                startedVariables = true;
              }
            } else if (startedVariables) {
            System.out.println("The format of a production is terminal, terminal ...
terminal, variable... varable. A terminal comes before the variable in line:");
            CurrentVariable.productions = null;
            //Clean out this baby to signal a problem.}
            }
          }
        }
        } while (wa.stopChar!= Globals.StoppingCharacter);

       if (LastAddedWord != null) {
       LastAddedWord.next = null;
       LastAddedWord = null;
    }

       if (! wa.doubleStop) {
    // At this point a production has been stored away. Now get its probability.}
        wa  = we.nextWord();
          if (wa.stopChar == Globals.EndOfFileSignifier) {
           System.out.println("Unexpected end of the file when looking for a
probability.");
```

```
            answer = false;
            return answer;
          }
           //System.out.print(":N:"+wa.w);
           TheProduction.setProbability(wordToValue(wa.w));

      }     else if (wa.stopChar != Globals.StoppingCharacter) {
            wa = we.nextWord();
            System.out.println("The word "'+ wa.w+'" does not belong in the
probability field. ");
    //   write('The error occurs in production:');
     //  PrintWordList(LastAddedWord)
    }
  }
  while ( ! wa.doubleStop);
  return answer;
}

   public BitNode InsertIntoBitList (BitNode node, BitNode ListStart){
    //This is just a temporary procedure
    // that maintains a list of BitNodes Sorted by Probability.
     BitNode Previous, Scanner;
    //For scanning along the list.
   BitNode answer=ListStart;
     if (ListStart ==null) {
       answer = node;
       node.up = null;
     } else if (node.probability <= ListStart.probability) {
       node.up = ListStart;
       answer = node;
     }  else {
       Scanner = ListStart;
       Previous = ListStart;
       while (Scanner != null) {
         if (node.probability > Scanner.probability){
            Previous = Scanner;
            Scanner = Scanner.up;
         } else {
             Previous.up = node;
            node.up = Scanner;
            Scanner = null;
            node = null;// To signal that it was added...
      }
    }
    if (node != null) {
         Previous.up = node;
         node.up = null;
      }
   }
   return answer;
 }

 public void BuildBitTable(){
 // It is important that there be a tree that
 // describes how the bits are assigned to each production.}

   BitNode IntermediateBitList;
    //  This contains the list of bit nodes that
    //  don't have a root. When there is only
    //  one left, then this is crowned king and
```

```
    //  assigned to the variable's ItsBitRoot.
   BitNode TempBit;
    // Used in the building.}
   ProductionNode productionList;
    // This is the list of productions that
    // the bit list will be built out of.
   int NextBitNumber;
    // This is used for assigning a unique
    // number to each node.
   VariableNode v;
    // This is so the list can do this
    // for every variable.

    v  = VariableListRoot;
    while (v != null) {
       productionList = v.productions;
       //This is the list.}
       IntermediateBitList = null;
       NextBitNumber = 0;
       while (productionList != null)  {
         TempBit = new BitNode();
         TempBit.setProbability(productionList.probability);
         TempBit.left = null;
         TempBit.right = null;
         TempBit.polarity = true;
         TempBit.bitNumber = -1;
         TempBit.theProductionNode = productionList;

         productionList.itsBit = TempBit;
         IntermediateBitList=InsertIntoBitList(TempBit, IntermediateBitList);
         productionList = productionList.next;
        }
    // Now one bit for each production list;
    // let's start making pairs.
    while (IntermediateBitList.up != null)  {
      //  While there is more than one node in the list.
       TempBit = new BitNode();
       TempBit.setProbability(IntermediateBitList.probability
                             + IntermediateBitList.up.probability);
       TempBit.left = IntermediateBitList;

       TempBit.right = IntermediateBitList.up;
       TempBit.bitNumber = NextBitNumber++;
       TempBit.polarity = true;
       TempBit.theProductionNode = null;

       IntermediateBitList = IntermediateBitList.up.up;
         // Get Rid of the top two.
       TempBit.left.polarity=false;
         // Flip one. One son should be true and the other false.
       TempBit.left.up = TempBit;
       TempBit.right.up = TempBit;
       IntermediateBitList=InsertIntoBitList(TempBit, IntermediateBitList);
  }
       // There should only be one left at this point.
     v.itsBitRoot = IntermediateBitList;
     IntermediateBitList.up = null;
     v = v.next;
 }
}
}
```

WordEater Class

```
import java.net.*;
import java.io.*;
public class WordEater {

  URL u;
  URLConnection uc;
  StringBuffer grammarBuffer;
  int grammarBufferPos=0;
  boolean readComplete=false;
  int numCharacters=-1;
    // For reading in the data.

  public final static int spacer=1;
  public final static int normal=2;
  public final static int stopper=3;

  public final static char equalityCharacter='=';
  public final static char stoppingCharacter='/';
  public final static char doubleStopCharacter=(char)(1);
    // This is sent out when a double slash is encountered.

  public int[] scanTable=new int[256];
    // This is a scan table used for parsing.

    public void initializeScanHashTable(){
    // The Scanning Hash Table has 256 entries.
    // They identify a character as either a letter, stop character,
    // a space or a comment.
    int i;
    for (i=14; i<=255; i++){
     scanTable[i]=normal;
  }
  for (i=0; i<=13; i++){
    scanTable[i]=spacer;
  }

      scanTable[(byte)(equalityCharacter)] = stopper;
       scanTable[(byte)(stoppingCharacter)] = stopper;
       scanTable[(byte)(' ')] = spacer;
}

public  void openGrammarURL(String a, URL u){
  // Takes a URL for a Grammar URL and reads it in line by line.
  // We just
  String line;

  if (a!="") {
        try{
          grammarBuffer = new StringBuffer();
          grammarBufferPos = 0;
          int pos=0;
          int len;
          u= new URL(u,a);
          uc=u.openConnection();
          DataInputStream theHTML = new DataInputStream(uc.getInputStream());
          while ((line= theHTML.readLine()) != null) {
              grammarBuffer.append(line);
            }
```

```
          grammarBuffer.append(' ');
        } catch (MalformedURLException e) {
          System.err.println(a+ " is a bad URL.");
          System.err.println(e);
        } catch (IOException e){
          System.err.println("IO Error reading in the data.");
        }
      }
  //  System.out.println("Hey, I found it.");
  }
public NextWordResponse nextWord () {
  String answer;
    // What goes back. Null if eof.
  answer = "";
  boolean ds=false;

  while ((grammarBufferPos<grammarBuffer.length())
      && (scanTable[grammarBuffer.charAt(grammarBufferPos)]==spacer)){
    grammarBufferPos++;
  }

  while ((grammarBufferPos<grammarBuffer.length())
     && (scanTable[grammarBuffer.charAt(grammarBufferPos)]==normal)){
    answer=answer+grammarBuffer.charAt(grammarBufferPos++);
  }
  answer=answer+" ";

    //System.out.println("Found:"+answer+":");


    if (grammarBufferPos>=grammarBuffer.length()){
      return new NextWordResponse(answer,Globals.EndOfFileSignifier,false);
    } else if (scanTable[grammarBuffer.charAt(grammarBufferPos)]==spacer) {
      grammarBufferPos++;
      return new NextWordResponse(answer,' ',false);
    } else {
      if ((scanTable[grammarBuffer.charAt(grammarBufferPos)]==stopper) &&
      (scanTable[grammarBuffer.charAt(grammarBufferPos-1)]==stopper)) {
        ds=true;
      }
      grammarBufferPos++;
      return new NextWordResponse(answer,grammarBuffer.charAt(grammarBufferPos-1),ds);
    }

  }


}
```

附录B 棒球CFG

此附录中包含一个与上下文无关的语法取样，它是被设计用来模仿一场棒球比赛的画外音。这个语法可用在附录A里的程序中所需要的格式打印出来。

每一个变量行都是以星号“*”开始。结果有两个部分，在它们前面加斜线隔开。第一部分是变量的混合和将在结果产生过程中代替受怀疑变量的终端。第二部分是在随机选择过程中给定的一个结果加权值。最后一部分是每一个变量行的结果，它是以两个斜线结尾的。

```
*period = ./1//
*questionmark = ?/1//
*WeatherComment = Hmm . Do you think it will rain ? /.1/
      What are the chances of rain today ? /.1/
      Nice weather as long as it doesn't rain . /.1/
      Well, if rain breaks out it
       will certainly change things . /.1/
      You can really tell the mettle
       of a manager when rain is threatened . /.1//
*BlogsOutfielder = Orville Baskethands /.1/
                   Robert Liddlekopf /.1/
                   Harrison "Harry" Hanihan /.1//
*BlogsInfielder = Gerry Johnson /.1/
                  Lefty Clemson /.1/
                  Robby Rawhide /.1/
                  Alberto Juan Turbosino /.1//
*BlogsManager = Billy Martin /.1/
                Hanson Haversham /.1//
*BlogsCatcher = Bloaty Von Ripple /.1//
*BlogsPitcher = Mark Markinson /.1/
                Andy Anteriority /.1//
*BlogsPlayer= Orville Baskethands /.1/
              Robert Liddlekopf /.1/
              Harrison "Harry" Hanihan /.1/
              Gerry Johnson /.1/
              Lefty Clemson /.1/
              Robby Rawhide /.1/
              Alberto Juan Turbosino /.1/
              Bloaty Von Ripple /.1/
              Mark Markinson /.1/
              Andy Anteriority /.1//
*WhapperOutfielder = Prince Albert von Carmichael /.1/
                     Parry Posteriority /.1/
                     Herbert Herbertson /.1//
*WhapperInfielder = Johnny Johanesberger /.1/
                    Frank Gavi /.1/
                    Harry Dolcetto /.1/
                    Sal Sauvignon /.1//
*WhapperManager = Bob Von Bittle /.1/
                  Hank Von Bittle /.1//
*WhapperCatcher = Mark Cloud /.1//
*WhapperPitcher = Jerry Johnstone /.1/
                  Albert Ancien-Regime /.1//
*WhapperPlayer = Prince Albert von Carmichael /.1/
```

```
                Parry Posteriority /.1/
                Herbert Herbertson /.1/
                Johnny Johanesberger /.1/
                Frank Gavi /.1/
                Harry Dolcetto /.1/
                Sal Sauvignon /.1/
                Jerry Johnstone /.1/
                Albert Ancien-Regime /.1/
                Mark Cloud /.1//
*Announcer- Bob /.1/
            Ted /.1/
            Mike /.1/
            Rich /.1//
*DumbComment - Some kind of Ballplayer, huh ? /.1/
         These guys came to play ball ./.1/
         What a game so far today ./.1/
         How about those players ./.1/
         Got to love baseball ./.1/
         Hey, they're playing the organ ./.1//
*WhapperOutfieldOut - He pops one up into deep left field . /.1/
          He lifts it back toward the wall where it is caught
          by *BlogsOutfielder *period/.1/
          He knocks it into the glove of
           *BlogsOutfielder *period /.1/
          He gets a real piece of it and
           drives it toward the wall
           where it is almost ... Oh My God! ... saved by
           *BlogsOutfielder *period /.1/
          He pops it up to *BlogsOutfielder *period /.2//
*WeirdOutfield - who is too deep to get it. Base hit! /.1/
     who bobbles the catch! /.05/
     who trips on his Nikes! Time for a new sponsor. /.05/
     who loses it in the sun. It drops in the warning track. /.1/
     whose mind seems lost thinking of investments. He starts
     moving too late! Hit. /.1//
*OutfieldResult - for a double! /.1/ for a stand-up double. /.1/
      for a stand-up double ... wait he's going to stretch
      it for a triple and ... he's safe. /.1/
      to grab another hit . /.1/
      to bump his average up . His salary is up for renegotiation
      this year . /.1//
*WhapperOutfieldHit - knocks it toward *BlogsOutfielder
          *WeirdOutfield /.1/
          line-drives toward the outfield where *BlogsOutfielder
          *WeirdOutfield /.1/
          puts it into the back corner of right field. /.1/
          lifts it over the head of *BlogsOutfielder
          *OutfieldResult /.1/
          drives it into the stands for a home run! /.1/
          lifts it toward heaven ! Home Run !/.1/
          sends it to the man upstairs . Home run !/.1/
          clears the stadium wall . Whoa ! home run !/.1/
          and it's ... La Bomba ! Home Run !/.1//
*BlogsOutfieldHit - knocks it toward *WhapperOutfielder
          *WeirdOutfield /.1/
          line-drives toward the outfield where *WhapperOutfielder
          *WeirdOutfield /.1/
          puts it into the back corner of right field. /.1/
          lifts it over the head of *WhapperOutfielder
          *OutfieldResult /.1/
          drives it into the stands for a home run! /.1/
```

```
          lifts it toward heaven ! /.1/
          sends it to the man upstairs . /.1/
          clears the stadium wall . Whoa ! /.1/
          and it's ... La Bomba ! /.1//
*BlogsOutfieldOut - He pops one up into deep left field . /.1/
          He lifts it back toward the wall where it is caught
          by *WhapperOutfielder *period/.1/
          He knocks it into the glove of
           *WhapperOutfielder *period /.1/
          He gets a real piece of it and
           drives it toward the wall
          where it is almost ... Oh My God! ... saved by
          *WhapperOutfielder *period /.1/
          He pops it up to *WhapperOutfielder *period /.2//
*BuntResponseHit - bobbles it ./.1/
    trips on it . /.1/
    scoops it up and realizing that he's too late
    signs it and throws it to a kid in the stands . / .01 /
    gets to, but the throw is ... late! /.1/
    grabs it, but the throw is too low ! /.1//
*BuntResponseOut - grabs it and makes the out . /.1/
    scoops it up and tosses him out . /.1/
    passes it on to first . /.1/
    nabs it ./.1/
    picks it up, laughs and tosses him out . /.1/
    grabs it and tosses it to first . /.1//
*InfieldLocation - first base /.1/
                   second base/.1/
                   the pitcher's mound/.1/
               short stop/.1/
               third base/.1/
               third base foul line/.1/
               first base foul line/.1//
*BadThrow - in the dirt ./.1/
            digging a ditch ./.1/
            too wide ./.1/
            into the umpire's head ! Whoa ! /.01/
            way too wide to make the tag ./.1/
          too high ./.1/
          too high to make the tag ./.1/
          in his chest ./.1//
*BaseReached - for a single ./.1/
               for a double ./.1/
               for a triple ./.1/
               for an easy single ...
                Wait he's going to try for more and the
                throw is *BadThrow /.1/
                for a stand-up double. But is that young
                Charlie Hustle going to wait ?
                No . He's going for the triple and
                the throw is *BadThrow /.1/
                for a close single and the the toss is
                *BadThrow /.1/
               for a dangerous double and the
                throw is *BadThrow /.1//
*HitResult - into short right field *BaseReached /.1/
       into short left field *BaseReached / .1/
       and the ball bounces past the shortstop *BaseReached/.1/
       and the ball takes a weird bounce at *InfieldLocation
       *BaseReached /.1/
```

```
      and the ball flies down the foul line .
       It stays fair *BaseReached /.1/
      in the short left field where it
      bounces into the stands . Ground rule
       double for the young man ./.1//
*WhapperInfieldHit - He tries to bunt, and *BlogsInfielder
        *BuntResponseHit /.1/
        He knocks it down the line between
        the legs of *BlogsInfielder *period /.1/
        He waps it into the shortstop's glove,
        but he can't control it . Safe at first . /.1/
        He lifts it over the head of *BlogsInfielder
         *HitResult /.1/
        The batter gets a piece of it *HitResult /.1/
        It's contact time  *HitResult /.1/
        Nice hit *HitResult /.1/
        Whoa ! That swing was on the money  *HitResult /.1/
        Nice job *HitResult /.1/
        Great hit *HitResult /.1/
        Super looper for a hit ./.1/
        He knocks a line-drive into the head of
         *BlogsInfielder /.05//
*WhapperInfieldOut - He grounds out to *BlogsInfielder *period/.1/
        He pops it up to *BlogsInfielder *period/.1/
        He tries to bunt, and *BlogsInfielder
         *BuntResponseOut /.1/
        He knocks a line-drive into the glove of
         *BlogsInfielder /.1/
        He knocks an easy bouncer to *BlogsInfielder
         *period /.1/
        He bounces one off the ground into the
         first-baseman's glove ./ .1//
*BlogsInfieldHit - He tries to bunt, and *WhapperInfielder
        *BuntResponseHit /.1/
        He knocks it down the line between the legs of
         *WhapperInfielder *period /.1/
        He lifts it over the head of *WhapperInfielder
         *HitResult /.1/
        The batter gets a piece of it *HitResult /.1/
        Nice job *HitResult /.1/
        Great hit *HitResult /.1/
        It's contact time  *HitResult /.1/
        Nice hit *HitResult /.1/
        He waps it into the shortstop's glove,
        but he can't control it . Safe at first . /.1/
        Whoa ! That swing was on the money  *HitResult /.1/
        Super looper for a hit ./.1/
        He knocks a line-drive into the head of
         *WhapperInfielder /.05//
*BlogsInfieldOut -
       He grounds out to *WhapperInfielder *period/.1/
       He  knocks a line-drive into the glove of
        *WhapperInfielder /.1/
       He tries to bunt, and
        *WhapperInfielder *BuntResponseOut /.1/
       He knocks an easy bouncer to
        *WhapperInfielder *period /.1/
       He  pops it up to *WhapperInfielder *period/.1/
       He  bounces one off the ground into the
        first-baseman's glove ./ .1//
```

```
*EndOfInning =
    Well, that's the end of their chances in
    this inning, *Announcer *period *Commercial/.1/
    No more at bats left in this inning. *Commercial /.1/
    The inning's over . Some kind of ballplayer, huh,
     *Announcer *period *Commercial /.1/
    Yowza! End of the inning . Hard to imagine
     life without baseball? Right, *Announcer
     *questionmark *Commercial /.1/
    That inning proves why baseball is
     the nation's game . *Commercial /.1//
*Return = Back to the game, *Announcer *period /.1/
    We now return to the game between the Blogs and
     the Whappers . /.1/
    Now back to the game . /.1/
    Just wanted to say thanks to our sponsors for
     those great messages . /.1//
*Commercial = This is WZZZ-TV bringing you the ballgame!
        *BeerCom *BeerCom
        *CarCom *Return/.1/
       We're here at WZZZ-TV bringing you the ballgame!
        *CarCom *CarCom
        *BeerCom *Return /.1/
       This is the WZZZ baseball network!
        *BeerCom *CarCom *CarCom
        *BeerCom *BeerCom *Return /.1/
       Now a message from our sponsors. *BeerCom *BeerCom
        *BeerCom *Return /.1/
       Now a very special message from our sponsors.
        *CarCom *BeerCom
        *CarCom *Return /.1//
*BeerOne = Imported Name . / .1/
           Imported Label ./.1/
           Imported Aura . /.1/
           Imported Concept ./.1/
           Imported Danger Warning in German . /.1//
*BeerTwo = American Taste . /.1/
           American Flavor . /.1/
           Clean American Taste ./.1/
           No Overly-flavorful Assault on your Tongue . /.1//
*BeerThree = fermentation is natural ? /.1/
       yeast is one of Mother Nature's Creatures ? /.1/
       yeast is a beast of Nature ? /.1/
       beer is natural ? /.1//
*BeerFour = Support Mother Nature and drink. /.1/
      Go green and support the environment . /.1/
      Support Natural Things and drink another ! /.1/
      Live a natural life and drink some more ! /.1//
*BeerCom = St. Belch . *BeerOne *BeerTwo /.1/
     Longing for adventure ? Open a bottle of St. Belch ! /.1/
     Did you know that *BeerThree *BeerFour /.1/
     Hey! *BeerFour /.1/
     Man comes up to Bob on the street and says,
     "Want a St. Belch ?" and his friend says, "Sure, I'm a man
     and I love to drink ." It turns out the Man was the head of
     a Fortune 500 company looking for a new Chairman of the
     Board of Directors . He hires Bob for
     $350,000 a year . /.1/
     St. Belch Beer: Especially tailored for men who watch
     ballgames . /.1//
```

```
*Car - Chevy Cordon-Bleu / .1 /
       Chevy Coq-au-Van /.1/
       Ford Platzer /.1/
       Ford Wienerschnitzel / .1/
       Chevy Choucroute /.1/
       Ford Weisswurst /.1/
       Chevy Crouton /.1//
*CarFoodAdj - Scrumptious ! /.1/
              Delicious ! /.1/
              Mouth-watering !/.1/
              Tasty! /.1/ Rich ! /.1/
              Sinfully Sweet ! /.1/
              Organically Grown/.1//
*CarRegAdj - Fast ! / .1/
             Bold ! / .1/ Ambitious ! /.1/
             Nocturnal ! /.1/ Adiabatic ! /.1/
             Anti-establishmentary !/.1/ Plenary ! /.1/
             Easy ! /.1/ Capitalistic ! /.1//
*CarLink - That's the new /.1/
           What a car, the /.1/
           How about that /.1/
           That's it. The new /.1//
*CarCom - Wow! *CarFoodAdj *CarRegAdj *CarFoodAdj *CarRegAdj
     *CarLink *Car *period /.1/
    What do these words mean? *CarRegAdj *CarFoodAdj *CarRegAdj
     *CarLink *Car *period /.1/
    Yowza! *CarFoodAdj *CarRegAdj *CarFoodAdj *CarLink
     *Car *period /.1/
    The way to a man's heart is through his stomach. Ergo we want
     you to think of our car as *CarFoodAdj *period *CarLink *Car
     *period /.1//
*PlateAction - comes to the plate ./.1/
         swings the bat to get ready and enters
          the batter's box ./.1/
         adjusts the cup and enters the batter's box ./.1/
         swings the baseball bat to stretch and enters
          the batter's box ./.1//
*NewBlogsBatter - Now, *BlogsPlayer *PlateAction /.1/
      Here we go. *BlogsPlayer *PlateAction /.1/
      The pitcher spits. *BlogsPlayer *PlateAction /.1/
      The crowd is nervous. *BlogsPlayer *PlateAction /.1//
*NewWhapperBatter - Now, *WhapperPlayer *PlateAction /.1/
      Here we go. *WhapperPlayer *PlateAction /.1/
      The pitcher spits. *WhapperPlayer *PlateAction /.1/
      The crowd is nervous. *WhapperPlayer *PlateAction /.1//
*Strike - Swings and misses ! /.1/ Fans the air ! /.1/
    No contact in Mudsville ! /.1/
    Whoooosh! Strike ! /.1/ Steeerllilike ! /.1/
    No wood on that one . /.1/
    He just watched it go by ./.1/ No contact on that one . /.1/
    Nothing on that one ./.1/
    He swings for the stands, but no contact . /.1//
*BallCall - Ball ./.1/ The umpire calls a ball ./.1/
      Definitely a ball ./.1/
      No strike this time . /.1//
*Ball - High and outside . *BallCall /.1/
        Whoa, he's brushing him back . *BallCall/.1/
        Short and away . *BallCall/.1/
        OOOh, that's almost in the dirt . *BallCall/.1/
        No good. *BallCall /.1/
        High and too inside. *BallCall/.1//
```

```
*ThePitchType = a curveball ./.1/
     a flaming fastball ./.1/
     a screaming fast ball ./.1/
     a change-up ./.1/
     a knuckler ./.1/
     what looks like a spitball ./.1/
     a spitter ./.1/
     a split-fingered fastball ./.1/ a rising fast ball ./.1/
     a fast one that looked like it was rising ./.1/
     a screamer ./.1/
     a torcher ./.1/ a blaster ./.1/a knuckleball ./.1/
     a breaking curve ./.1/
     a slow change-up ./.1/ a toaster ./.1/
     a rattling corkscrew ./.1/
     a curvaceous beauty ./.1/
     a wobbling knuckler ./.1/
     a bouncing knuckleball ./.1/
     a smoking gun ./.1/ a blazing comet ./.1/
     a rattler ./.1/
     a heckraising fastball ./.1/
     a rocket booster ./.1/ a fastball with wings ./.1//
*ThePitch = Checks first base . Nothing. Winds up and pitches
      *ThePitchType /.1/
      Here's the pitch . It's *ThePitchType /.1/
      Here comes the pitch . It's *ThePitchType/.1/
      He's winding up . What *ThePitchType /.1/
      And the next pitch is *ThePitchType /.1/
      The next pitch is *ThePitchType /.1/
      A full windup and it's *ThePitchType /.1/
      He's uncorking *ThePitchType /.1/
      It's *ThePitchType /.1//
*StrikeOut = He's out of there . / .1/
       Strike out . He's swinging at the umpire .
       The umpire reconsiders until
       the security guards arrive . /.1/
       Strike out ! /.1/
       Yes. Another wiffer ./.1/
       Strike out . There goes his batting average . /.1/
       Strike three . Some sports writer figured that
       each strike cost the ball
       player about $1,530 at today's average rate . /.1/
       The last strike . Only three chances in this game . /.1//
*BlogsHit = Here we go. *ThePitch *Ball *ThePitch *BlogsInfieldHit /.1/
      Okay. *ThePitch *Ball *Strike *Strike *BlogsInfieldHit /.1/
      The crowd is roaring . *ThePitch *Ball *ThePitch
       *Ball *BlogsOutfieldHit/.1/
      Here we go . *Ball *ThePitch *Ball
       *ThePitch *BlogsOutfieldHit /.1/
      Here's the pitch . *Strike *ThePitch *Strike *ThePitch
       *BlogsOutfieldHit /.1//
*BlogsOut = Yeah. *ThePitch *Strike *ThePitch *Strike *ThePitch
      *Strike *StrikeOut /.1/
      The pitcher is winding up to throw. *Strike
       *ThePitch *Ball *ThePitch
       *BlogsInfieldOut /.1/
       Here's the fastball . *BlogsInfieldOut /.1/
       He's trying the curveball . *BlogsOutfieldOut /.1/
       Love that baseball game. *ThePitch *Ball
        *ThePitch *Ball *ThePitch
       *Strike *ThePitch *Strike *ThePitch *Strike *StrikeOut /.1/
```

```
      Another fastball . *Strike *ThePitch
       *BlogsOutfieldOut /.1//
*WhapperHit - Here we go. *ThePitch *Ball *ThePitch
      *WhapperInfieldHit /.1/
      Okay. *ThePitch *Ball *Strike *Strike *WhapperInfieldHit /.1/
      The crowd is roaring . *ThePitch *Ball *ThePitch
      *Ball *WhapperOutfieldHit/.1/
      Here we go . *Ball *ThePitch *Ball
       *ThePitch *WhapperOutfieldHit /.1/
      Here's the pitch . *Strike *ThePitch
       *Strike *ThePitch
       *WhapperOutfieldHit /.1//
*WhapperOut - Yeah. *ThePitch *Strike *ThePitch *Strike
      *ThePitch *Strike *StrikeOut /.1/
      The pitcher is winding up to throw. *Strike *ThePitch *Ball
      *ThePitch
      *WhapperInfieldOut /.1/
      Here's the fastball . *WhapperInfieldOut /.1/
      He's trying the curveball . *WhapperOutfieldOut /.1/
      Love that baseball game. *ThePitch *Ball
       *ThePitch *Ball *ThePitch
       *Strike *ThePitch *Strike *ThePitch
       *Strike *StrikeOut /.1/
      Another fastball . *Strike *ThePitch
       *WhapperOutfieldOut /.1//
*OneWhapperLeft - Only one more out needed . *NewWhapperBatter
    *WhapperHit
    *OneWhapperLeft /.1/
    The Blogs only need to get one more out . *NewWhapperBatter
    *WhapperHit *OneWhapperLeft /.1/
    The Blogs are trying for the last out .  *NewWhapperBatter
    *WhapperHit *OneWhapperLeft /.1/
    The crowd is looking for the last out !   *NewWhapperBatter
    *WhapperHit *OneWhapperLeft /.1/
    Before the last out, let's get a message in from our
    sponsors. *Commercial   *NewWhapperBatter
    *WhapperHit *OneWhapperLeft /.05/
    Only one out left.   *NewWhapperBatter
    *WhapperHit *OneWhapperLeft /.1/
    Yup, the Blogs only need to get one more out.
     *NewWhapperBatter
    *WhapperHit *OneWhapperLeft /.1/
    Hey, two down, one to go. *NewWhapperBatter
    *WhapperOut *EndOfInning /.1/
    Only one chance left for the Whappers . *NewWhapperBatter
    *WhapperOut *EndOfInning /.1/
    Two big outs for the Blogs. *NewWhapperBatter
    *WhapperOut *EndOfInning /.1/
    What a game ! *NewWhapperBatter
    *WhapperOut *EndOfInning /.1/
    These are the times that make baseball special .
      *NewWhapperBatter
    *WhapperOut *EndOfInning /.1/
    One more thin out stands between the Whappers and the end of
    this inning's chances . *NewWhapperBatter
    *WhapperOut *EndOfInning /.1/
    Yowza. *NewWhapperBatter
    *WhapperOut *EndOfInning /.1/
    He's hefting some wood . *NewWhapperBatter
    *WhapperOut *EndOfInning /.1//
```

```
*TwoWhappersLeft - Hey, one down, two to go.   *NewWhapperBatter
       *WhapperHit *TwoWhappersLeft /.1/
       The Whappers have only one out .       *NewWhapperBatter
       *WhapperHit *TwoWhappersLeft /.1/
       One down, two to go.  *NewWhapperBatter
       *WhapperHit *TwoWhappersLeft /.1/
       Only one out into the inning.  *NewWhapperBatter
       *WhapperHit *TwoWhappersLeft /.1/
       One out against the Whappers.  *NewWhapperBatter
       *WhapperHit *TwoWhappersLeft /.1/
       The Blogs need two more outs. *NewWhapperBatter
       *WhapperOut *OneWhapperLeft /.1/
       Two more outs to go. *NewWhapperBatter
       *WhapperOut *OneWhapperLeft /.1/
       The Whappers have two outs to spare. *NewWhapperBatter
       *WhapperOut *OneWhapperLeft /.1/
       Plenty of room. Only one out. *NewWhapperBatter
       *WhapperOut *OneWhapperLeft /.1/
       How about those ballplayers.
        One out so far. *NewWhapperBatter
       *WhapperOut *OneWhapperLeft /.1/
       Some day for a ballgame, huh? *WeatherComment *NewWhapperBatter
       *WhapperOut *OneWhapperLeft /.1/
       Wow. Only one out. *NewWhapperBatter
       *WhapperOut *OneWhapperLeft /.1/
       Somekind of ballgame, huh, *Announcer
        *questionmark *NewWhapperBatter
       *WhapperOut *OneWhapperLeft /.1/
       Yup, got to love this stadium. *NewWhapperBatter
        *WhapperOut *OneWhapperLeft /.1//
*ThreeWhappersLeft - No damage yet. *Announcer *period
        *NewWhapperBatter
        *WhapperHit *ThreeWhappersLeft /.1/
        No outs . *NewWhapperBatter
        *WhapperHit *ThreeWhappersLeft /.1/
        No outs yet for the Whappers . *NewWhapperBatter
        *WhapperHit *ThreeWhappersLeft /.1/
        No trouble yet . *NewWhapperBatter
        *WhapperHit *ThreeWhappersLeft /.1/
        Check out the Whappers' mascot a Weasel.
         He's biting a 7-year old !
         Some kind of mascot, huh, *Announcer *questionmark
         *NewWhapperBatter
        *WhapperOut *TwoWhappersLeft /.01/
        Nobody out yet . *NewWhapperBatter
        *WhapperOut *TwoWhappersLeft /.1/
        Whappers are up with no outs . *NewWhapperBatter
        *WhapperOut *TwoWhappersLeft /.1/
        Nothing's happened yet to the Whappers . *NewWhapperBatter
        *WhapperOut *TwoWhappersLeft /.1/
        No outs yet . *NewWhapperBatter
        *WhapperOut *TwoWhappersLeft /.1/
        What a day . *NewWhapperBatter
        *WhapperOut *TwoWhappersLeft /.1/
        I could go for a home run.
         *Announcer *period *NewWhapperBatter
        *WhapperOut *TwoWhappersLeft /.1/
        Baseball and Apple Pie . *NewWhapperBatter
        *WhapperOut *TwoWhappersLeft /.1/
        Yup.  *DumbComment *NewWhapperBatter
        *WhapperOut *TwoWhappersLeft /.2//
```

```
*OneBlogsLeft - Only one more out needed . *NewBlogsBatter *BlogsHit
    *OneBlogsLeft /.1/
    The Whappers only need to get one more out . *NewBlogsBatter
    *BlogsHit *OneBlogsLeft /.1/
    The Whappers are trying for the last out .  *NewBlogsBatter
    *BlogsHit *OneBlogsLeft /.1/
    The crowd is looking for the last out !   *NewBlogsBatter
    *BlogsHit *OneBlogsLeft /.1/
    Before the last out, let's get a message in from our
    sponsors. *Commercial   *NewBlogsBatter
    *BlogsHit *OneBlogsLeft /.05/
    Only one out left.   *NewBlogsBatter
    *BlogsHit *OneBlogsLeft /.1/
    Yup, the Whappers only need to get
     one more out.   *NewBlogsBatter
    *BlogsHit *OneBlogsLeft /.1/
    Hey, two down, one to go. *NewBlogsBatter
    *BlogsOut *EndOfInning /.1/
    Only one chance left for the Blogs . *NewBlogsBatter
    *BlogsOut *EndOfInning /.1/
    Two big outs for the Whappers. *NewBlogsBatter
    *BlogsOut *EndOfInning /.1/
    What a game ! *NewBlogsBatter
    *BlogsOut *EndOfInning /.1/
    These are the times that make baseball
     special . *NewBlogsBatter
    *BlogsOut *EndOfInning /.1/
    One more thin out stands between the Blogs and the end of
    this inning's chances . *NewBlogsBatter
    *BlogsOut *EndOfInning /.1/
    Yowza. *NewBlogsBatter
    *BlogsOut *EndOfInning /.1/
    He's hefting some wood . *NewBlogsBatter
    *BlogsOut *EndOfInning /.1//
*TwoBlogsLeft - Hey, one down, two to go.   *NewBlogsBatter
      *BlogsHit *TwoBlogsLeft /.1/
      The Blogs have only one out .      *NewBlogsBatter
      *BlogsHit *TwoBlogsLeft /.1/
      One down, two to go.  *NewBlogsBatter
      *BlogsHit *TwoBlogsLeft /.1/
      Only one out into the inning.  *NewBlogsBatter
      *BlogsHit *TwoBlogsLeft /.1/
      One out against the Blogs.  *NewBlogsBatter
      *BlogsHit *TwoBlogsLeft /.1/
      The Whappers need two more outs. *NewBlogsBatter
      *BlogsOut *OneBlogsLeft /.1/
      Two more outs to go. *NewBlogsBatter
      *BlogsOut *OneBlogsLeft /.1/
      The Blogs have two outs to spare. *NewBlogsBatter
      *BlogsOut *OneBlogsLeft /.1/
      Plenty of room. Only one out. *NewBlogsBatter
      *BlogsOut *OneBlogsLeft /.1/
      How about those ballplayers. One out so
       far. *NewBlogsBatter
      *BlogsOut *OneBlogsLeft /.1/
      Some day for a ballgame, huh?
       *WeatherComment *NewBlogsBatter
      *BlogsOut *OneBlogsLeft /.1/
      Wow. Only one out. *NewBlogsBatter
      *BlogsOut *OneBlogsLeft /.1/
```

```
       Some kind of ballgame, huh, *Announcer
        *questionmark *NewBlogsBatter
       *BlogsOut *OneBlogsLeft /.1/
       Yup, got to love this stadium. *NewBlogsBatter
       *BlogsOut *OneBlogsLeft /.1//
*ThreeBlogsLeft - No damage yet. *Announcer *period *NewBlogsBatter
        *BlogsHit *ThreeBlogsLeft /.1/
        No outs . *NewBlogsBatter
        *BlogsHit *ThreeBlogsLeft /.1/
        No outs yet for the Blogs . *NewBlogsBatter
        *BlogsHit *ThreeBlogsLeft /.1/
        No trouble yet . *NewBlogsBatter
        *BlogsHit *ThreeBlogsLeft /.1/
        Check out the Blogs' Weasel. He's biting a 7-year old !
        Some kind of mascot, huh, *Announcer *questionmark
        *NewBlogsBatter
        *BlogsOut *TwoBlogsLeft /.01/
        Nobody out yet . *NewBlogsBatter
        *BlogsOut *TwoBlogsLeft /.1/
        Blogs are up with no outs . *NewBlogsBatter
        *BlogsOut *TwoBlogsLeft /.1/
        Nothing's happened yet to the Blogs .
         *NewBlogsBatter
        *BlogsOut *TwoBlogsLeft /.1/
        No outs yet . *NewBlogsBatter
        *BlogsOut *TwoBlogsLeft /.1/
        What a day . *NewBlogsBatter
        *BlogsOut *TwoBlogsLeft /.1/
        I could go for a home run.
         *Announcer *period *NewBlogsBatter
        *BlogsOut *TwoBlogsLeft /.1/
        Baseball and Apple Pie . *NewBlogsBatter
        *BlogsOut *TwoBlogsLeft /.1/
        Yup.  *DumbComment *NewBlogsBatter
        *BlogsOut *TwoBlogsLeft /.2//
*BlogsHalf - Time for the Blogs to see what they can do .
       *ThreeBlogsLeft /.1/
       It's the Blogs' turn at bat . *ThreeBlogsLeft /.1/
       Bottom half of the inning .
       The Blogs must prove their stuff .
       *ThreeBlogsLeft /.1/
       Let's get on with the inning . *ThreeBlogsLeft /.1/
       Well, another half of the inning, *Announcer *period
       *DumbComment
       *ThreeBlogsLeft /.1/
       Let's get moving  with this half of the inning, but first,
       another message from our sponsors . *Commercial
       *ThreeBlogsLeft /.1//
*WhapperHalf - Top of the inning. *ThreeWhappersLeft /.1/
    Another new inning . Ain't life great, *Announcer *questionmark
    *ThreeWhappersLeft /.1/
    Start of another inning . *ThreeWhappersLeft /.1/
    Yes, it's time for the Whappers to lead off the inning .
    *ThreeWhappersLeft /.1/
    Time for another inning . The Whappers will be leading off .
    *ThreeWhappersLeft /.1//
*NineInnings - Let's get going ! *WhapperHalf *BlogsHalf
         *WhapperHalf *BlogsHalf
         *WhapperHalf *BlogsHalf
         *WhapperHalf *BlogsHalf
```

```
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf /.1/
        The Umpire throws out the ball . *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf /.1/
        Play ball ! *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf /.1/
        It's a fine day for a game . *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf
        *WhapperHalf *BlogsHalf /.1//
*OneInning = Let's get going !
            *WhapperHalf *BlogsHalf/.1/
            The umpire throws out the ball .
            *WhapperHalf *BlogsHalf/.1/
            Play ball ! *WhapperHalf *BlogsHalf/.1/
           It's a fine day for a game . *WhapperHalf
             *BlogsHalf/.1//
*AAStart = Well Bob, welcome to yet another game between the
     Whappers and the Blogs here in scenic downtown
     Blovonia . I think it is fair to say that there is
     plenty of BlogFever brewing in the stands as the
     hometown comes out to root for its favorites .
      *OneInning /.1/
     It's time for another game between the Whappers and
     the Blogs in scenic downtown Blovonia . I've just got
     to say that the Blog fans have come to support their
     team and rant and rave . *OneInning /.1//
```

附录C 可逆语法生成器

这里是第8章描述的可逆语法生成器的LISP源代码。它被写为XLISP，一个能在整个国际互联网的很多存档里找到的LISP译本。虽然其结构是很基本的，但它还是很有用的。

```
;;; Reversible Grammar Machine
;;; Copyright 1996 Peter Wayner
;;; All rights reserved.
;;; Permission is granted to copy the file as long as no charge
;;; is made. Permission is also granted to make changes as long as
;;; the author of the changes is indicated in the comments.
;;;
;;;
;;; This code is designed to implement a reversible computer. If
;;; it can be reversed, then the data used to create it can be extracted.
;;;
;;;
;;; constant-list contains constant values that are left unchanged.
;;; var-list contains variables that are changed by the person.
;;; procedure-list includes all of the procedures that are executed.

(setq constant-list
        '(
                (c1 ("Bob " "Ray " "Loraine " "Carol " "Gilda "))
                (c2 ("Lucy " "Ricky " "Ethel " "Fred "))
                (c3 ("Fred " "Barney " "Wilma " "Betty "))

                (v1 ("considered insubordination "
                     "redirected commercial inertia "
                     "smiled knowingly "
                     "reundeconstructed comedic intent "
                     "ladled laugh slop "
                     "insinuated a nervous satire "))

               (cs 1) (ci 1) (ce 5)

        ))

(setq var-list
        '(
               (va 3) (vb 4) (vc 45) (vd 1) (ve 41) (vf 11)
        ))

(setq procedure-list
        '(
               (main ((add va vb)
                      (whi vf cs ci (gt vf ce) t1)
                      (chz (c1)) (chz (c1 c2)) (add vb vc)
                      (chz (v1)) (chz (c2 c3)) (mul vc va)
                      (chz (v1))))

               (t1          ((add va vb)
                      ;;(add vb vc) (mul vd va) (mul vc vd)
                      (if (gt va vb) b1 b2) (add vc vd) (add vd ve)))
```

```
                (b1   ((mul vd ve) (mul vb ve)))
                (b2   ((add vd vc) (add vc ve)))
        )
)

(defun Record-Error (error-string)
;;; If something goes wrong, this indicates the error.
        (print error-string))

(defun eval-tag (tag)
;;; Finds the right match for the tag.
        (setq temp nil)
        (setq temp (assoc tag constant-list))
        (cond ((not temp)
                        (setq temp (assoc tag var-list))))
        (cond ((not temp)
                        (Record-Error (concatenate 'string
                                                "Missing Tag:"
                                                (symbol-name tag)))))
        temp)
(defun test-change (tag new-value)
        (setq temp (assoc tag var-list))
        (cond (temp
                        (setf (cdr temp) new-value))))


(defun do-addition (tag1 tag2)
;; Add tag2+tag1 and store in tag1
        (setq temp (eval-tag tag2))
        (setq temp2 (assoc tag1 var-list))
        (cond ((not temp2)
                        (Record-Error (concatenate 'string
                                                "Missing Variable Tag:"
                                                (symbol-name tag1))))
                ((and temp (numberp (cadr temp))
                        (numberp (cadr temp2)))
                        (setf (cdr temp2)
                                (list (+ (cadr temp) (cadr temp2)))))
                (t
        (Record-Error (concatenate 'string
                                "Problems with adding:"
                                (symbol-name tag1)
                                (symbol-name tag2))))))

(defun do-subtraction (tag1 tag2)
;; Subtract tag1-tag2 and store in tag1
        (setq temp2 (eval-tag tag2))
        (setq temp1 (assoc tag1 var-list))
        (cond ((not temp1)
                        (Record-Error (concatenate 'string
                                                "Missing Variable Tag:"
                                                (symbol-name tag1))))
                ((and temp2 (numberp (cadr temp1))
                        (numberp (cadr temp2)))
                        (setf (cdr temp1)
                                (list (- (cadr temp1) (cadr temp2)))))
                (t
                  (Record-Error (concatenate 'string
                                "Problems with subtracting:"
                                (symbol-name tag1)
                                (symbol-name tag2))))))
```

```
(defun do-multiplication (tag1 tag2)
;; Add tag2+tag1 and store in tag1
       (setq temp (eval-tag tag2))
       (setq temp2 (assoc tag1 var-list))
       (cond ((not temp2)
                      (Record-Error (concatenate 'string
                                         "Missing Variable Tag:"
                                         (symbol-name tag1))))
                 ((and temp (numberp (cadr temp))
                           (numberp (cadr temp2)))
                 (cond ((or (= 0 (cadr temp2)) (= 0 (cadr temp)))
                                (Record-Error (concatenate 'string
                                         "Multiply by zero error: "
                                         (symbol-name tag1) " by "
                                         (symbol-name tag2))))
                           (t
                            (setf (cdr temp2)
                            (list (* (cadr temp) (cadr temp2)))))))
                 (t
                   (Record-Error (concatenate 'string
                               "Problems with multiplying: "
                               (symbol-name tag1) " by "
                               (symbol-name tag2))))))


(defun do-division (tag1 tag2)
;; Add tag2+tag1 and store in tag1
       (setq temp (eval-tag tag2))
       (setq temp2 (assoc tag1 var-list))
       (cond ((not temp2)
                      (Record-Error (concatenate 'string
                                         "Missing Variable Tag: "
                                         (symbol-name tag1))))
                 ((and temp (numberp (cadr temp))
                           (numberp (cadr temp2)))
                (cond ((or (= 0 (cadr temp2)) (= 0 (cadr temp)))
                               (Record-Error (concatenate 'string
                                         "Divide by zero error: "
                                         (symbol-name tag1) " by "
                                         (symbol-name tag2))))
                           (t
                            (setf (cdr temp2)
                            (list (/ (cadr temp2) (cadr temp)))))))
                 (t
                   (Record-Error (concatenate 'string
                               "Problems with Dividing: "
                               (symbol-name tag1) " by "
                               (symbol-name tag2))))))
(defun Do-Swap (var1 var2)
;;; Swap the values stored here.
       (setq temp (cdr (assoc var1 var-list)))
       (setf (cdr (assoc var1 var-list)) (cdr (assoc var2 var-list)))
       (setf (cdr (assoc var2 var-list)) temp))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; Eval Operations
;;;; Used to step through lists of ops.
(defvar Forbidden-List nil)
;;; This is the list of variables that can't be touched.
```

```
;;; This feature is used when progressing down paths chosen
;;; by an if statement. If the variable used to choose the
;;; path is changed along the path, then bad things can happen
;;; and the world can't be reversed correctly.
(defun check-if-test (if-test)
;;; Evaluate test.
 (setq ans nil)
   (setq side1 (cadr (eval-tag (cadr if-test))))
   (setq side2 (cadr (eval-tag (caddr if-test))))
   (cond ((eq (car if-test) 'lt) ;; less-than
          (setq ans (< side1 side2)))
         ((eq (car if-test) 'gt) ;; greater-than
          (setq ans (> side1 side2)))
                ((eq (car if-test) 'eq) ;; equal-than
          (setq ans (= side1 side2)))
         ((eq (car if-test) 'le) ;; less-or-equal
          (setq ans (<= side1 side2)))
         ((eq (car if-test) 'ge) ;; greater-or-equal
          (setq ans (>= side1 side2)))
         (t
          (Report-Error (concatenate 'string
                         "Error evaluating If-then:"
                          (symbol-name (car if-test))))))
         ans)
 (defun Do-If (if-test if-clause else-clause)
 ;;; Evaluate an if-then branch.
 ;;;
 ;;; This current version can CREATE bugs when
 ;;; the program takes one path and then CHANGES
 ;;; the value of one of the variables used to
 ;;; choose the path. Reversing this is filled
 ;;; with ambiguities and is prohibited.
 ;;;
         (setq ans (check-if-test if-test))
       (setq Forbidden-List (append
                             (list (cadr if-test) (caddr if-test))
                             Forbidden-List))
    (cond (ans
           (Do-Op-List if-clause))
          (t
           (Do-Op-List else-clause)))
    (setq Forbidden-List (cdr (cdr Forbidden-List)))
    )
 (defun Do-Reverse-If (if-test if-clause else-clause)
 ;;; Evaluate an if-then branch.
 ;;; This goes backward. I could remove some of this
 ;;; extra code by embedding a strategic reverse, but I
 ;;; think it might be better to build the separate code
 ;;; now.
   (setq ans (check-if-test if-test))
    (cond (debug
          (print (list "In Rev If. Taking path:" ans))))
    (cond (ans
           (Do-Reverse-Op-List if-clause))
          (t
           (Do-Reverse-Op-List else-clause))))
 (defun Do-While (var init-const inc-const if-test branch)
 ;;; A while operation looks like this: '(whi var init-const inc-const
 ;;; test branch). var is the variable that is used to keep track of the
 ;;; progress of the loop. The first pass through the loop, var is set
```

```
;;; to init-const. Then if-test is checked. When it becomes true, then
;;; the loop breaks out. Otherwise the branch is executed. Then
;;; inc-const is ADDed to var and we go back to if-test.
;;;
;;; There is something irreversible going on here!!! The prior contents
;;; of var are destroyed. This is only bad if you use the variable
;;; before it gets to the loop. The solution is to use a new variable
;;; for each loop. This is, alas, a requirement. Something must keep
;;; track of the number of passes through the loop.
        (setf (cdr (assoc var var-list))
              (append (cdr (assoc init-const constant-list)) nil))
        (setq ans (check-if-test if-test))
        (do () (ans)
            (Do-Op-List branch)
                (Do-Addition var inc-const)
                (setq ans (check-if-test if-test)))
)

(defun Do-Reverse-While (var init-const inc-const if-test branch)
;;; The while loop run in reverse!
        (do () ((= (cadr (assoc var var-list))
                   (cadr (assoc init-const constant-list))))
            (Do-Reverse-Op-List branch)
                (Do-Subtraction var inc-const)
        )
)



(defun Do-Operation (op)
;;; An operation looks like this: '(op var1 var2)
;;; Do the right thing.
        (cond (debug
                        (print (list "Doing Op:" op))))

        (cond ((member (cadr op) Forbidden-List)
               (Record-Error (concatenate 'string
                            "Attempt to influence op on Forbidden List:"
                            (symbol-name (cadr op)))))

              ((eq (car op) 'add)
               (Do-Addition (cadr op) (caddr op)))
              ((eq (car op) 'sub)
               (Do-Subtraction (cadr op) (caddr op)))
              ((eq (car op) 'mul)
               (Do-Multiplication (cadr op) (caddr op)))
              ((eq (car op) 'div)
               (Do-Division (cadr op) (caddr op)))
              ((eq (car op) 'swp)
               (Do-Swap (cadr op) (caddr op)))
              ((eq (car op) 'if)
               (Do-If (cadr op) (caddr op) (cadddr op)))
              ((eq (car op) 'chz)
               (Do-Choice (cadr op)))
              ((eq (car op) 'whi)
               (Do-While (cadr op) (caddr op) (cadddr op)
                         (nth 4 op) (nth 5 op)))
                 (t
                         (Record-Error (concatenate 'string
                                       "Undefined Operation: "
                                       (symbol-name (car op)))))
        ))
```

```
(defun Do-Reverse-Operation (op)
;;; An operation looks like this: '(op var1 var2)
;;; Do the right thing.
        (cond (debug
                        (print (list "Doing Reverse Op:" op))))
        (cond ((eq (car op) 'add)
               (Do-Subtraction (cadr op) (caddr op)))
              ((eq (car op) 'sub)
               (Do-Addition (cadr op) (caddr op)))
              ((eq (car op) 'mul)
                (Do-Division (cadr op) (caddr op)))
               ((eq (car op) 'div)
                (Do-Multiplication (cadr op) (caddr op)))
               ((eq (car op) 'swp)
                (Do-Swap (cadr op) (caddr op)))
               ((eq (car op) 'if)
                (Do-Reverse-If (cadr op) (caddr op) (cadddr op)))
                   ((eq (car op) 'chz)
                (Do-Reverse-Choice (cadr op)))
               ((eq (car op) 'whi)
                (Do-Reverse-While (cadr op)
                                  (caddr op) (cadddr op)
                                  (nth 4 op) (nth 5 op)))

               (t
                         (Record-Error (concatenate 'string
                                           "Undefined Operation: "
                                           (symbol-name (car op)))))
         ))

(defun Do-Op-List (ls)
;;; Look up the tag in the procedure list.
   (do ((l (cadr (assoc ls procedure-list)) (cdr l)))
       ((null l))
     (Do-Operation (car l))))
(defun Do-Reverse-Op-List (ls)
;;; Look up the tag in the procedure list.
   (do ((l (reverse (cadr (assoc ls procedure-list))) (cdr l)))
       ((null l))
     (Do-Reverse-Operation (car l))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; File Operations
;;;; Used for going backward.
(defvar Backwards-Text nil)
;;; This file is where the text will be removed.
(defvar Backwards-Buffer "")
;;; This holds the reversed version of the string.
(defun File-To-String (name)
;;; Reads in a file. Might be faster if AREF is used
;;; to index the string.
        (setq temp "")
        (setq Backwards-Text
          (open name))
        (setq a (read-char backwards-text))
        (do ()
                ((null (peek-char 'nil backwards-text)))
             (setq temp (concatenate 'string temp (string a)))
             (setq a (read-char backwards-text)))
        (setq temp (concatenate 'string temp (string a)))
        (close backwards-text)
        temp)
```

```
(defun File-To-BitString (name)
;;; Reads in a file and converts it into a bit string.
;;; Might be faster if AREF is used
;;; to index the string.
;;;
;;; This is a really inefficient way to do things. It
;;; is silly to load an entire file into memory. But,
;;; I'm getting lazy in these days of cheap memory.
;;; If this isn't changed to buffer things, then I
;;; didn't have time to be efficient.
        (setq temp "")
        (setq temp-file
          (open name))
        (setq a (read-byte temp-file))
        (do ()
                ((null (peek-char 'nil temp-file)))
              (setq temp (concatenate 'string temp (Num-To-Bits a 256)))
              (setq a (read-byte temp-file)))
        (close temp-file)
        (setq BitSourcePointer 0)
        (setq MaxBitSourcePointer (length temp))
        temp)

(defun BitString-To-File (str name)
;;; Take a bit string and convert it into bytes.
        (setq out-file (open name :direction :io ))
        (setq tot-bytes (ceiling (/ (length str) 8)))
        (setq ptr 0)
        (do ((i 0 (+ 1 i))) ((= tot-bytes i))
                (setq cur-byte 0)
                (setq cur-value 128)
                (do ((j 0 (+ 1 j))) ((= j 8))
                        (cond ((eq (aref str ptr) #)
                                (setq cur-byte (+ cur-byte cur-value))))
                        (setq cur-value (/ cur-value 2))
                        (setq ptr (+ 1 ptr)))
                (write-byte cur-byte out-file))
        (close out-file))


(defun write-string (st stream)
  (do ((i 0 (+ 1 i))) ((= i (length st)))
    (write-char (aref st i) stream)))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; Choose Operation and its reverse.
(defun Flatten-Choice-List (ls)
;;; ls is presented as a list of variables and constants.
;;; Some of these variables and constants might be lists
;;; of variables and constants.
;;; This current version is NOT recursive. It can
;;; handle ONE level of indirection.
        (setq answer nil)
        (do ((l ls (cdr l))) ((null l))
          (cond ((stringp (car l))
                  (setq answer (cons (car l) answer)))
```

```
                ((atom (car l))
                 (setq answer (append
                                   (cadr (eval-tag (car l)))
                                   answer)))
                (t
                 (Report-Error (concatenate 'string
                             "Something wrong with: "
                             (symbol-name (car l))))))
        answer)
(defun Stringify (i)
;;; Make sure that i is a string.
        (cond ((listp i)
               (Flatten-Choice-List i))
              ((stringp i) i)
              ((integerp i)
               (format nil "~d" i))
              ((floatp i)
               (format nil "~g" i))
              ((rationalp i)
               (format nil "~d" i))))

(defun Find-Reverse-Choice (ls)
;;; Scans down ls and looks for the first complete match with
;;; the Backwards-Buffer.
        (setq answer nil)
        (setq counter 0)
        (do ((l (car ls) (cdr l)))
                ((null l))
           (setq ttt (reverse (car l)))
           (cond ((string= ttt Backwards-Buffer :end2 (length ttt))
                   ;;; We have a match!!!
                   (setq answer counter)
                   (setq l nil)
                   )
                  (t (setq counter (+ 1 counter)))))
         answer)

(defun Num-To-Bits (value top)
;;; There are top choices between 0 and top-1.
;;; Find the bits to this value.
        (setq bot 0)
        (setq top (- top 1))
        (setq answer "")
        (do ()
                ((= top bot))
        ;;  (print (list bot top))
          (cond ((> value (+ (/  (- top bot) 2) bot))
                   (setq bot (ceiling (+ (/ (+ 1 (- top bot)) 2) bot)))
                   (setq answer (concatenate 'string answer "1")))
                  (t
                   (setq top (floor (+ (/  (- top bot) 2) bot)))
                   (setq answer (concatenate 'string answer "0")))))
         answer)
(defun test-num (j)
  (setq answer nil)
  (do ((i 0 (+ 1 i))) ((= i j))
    (setq temp (num-to-bits i j))
    (cond ((not (= i (Bits-To-Num temp j)))
           (print (list i (Bits-To-Num temp j) j))))))
(defun Bits-To-Num (bits top )
```

```
;;; Reverse Num-To-Bits.
;;; This consumes bits from the BitFile.
;;; Choose them at random if the file is finished.
    (setq bot 0)
    (setq top (- top 1))
        (do ((i BitSourcePointer (+ 1 i))) ((= top bot))
            (setq BitSourcePointer (+ 1 BitSourcePointer))
            (cond (debug
                     (print (list "taking bit: " (aref bits i) i))))
                (cond ((> BitSourcePointer MaxBitSourcePointer)
                       (cond ((= 0 (rand 1))
                        (setq top (floor (+ (/  (- top bot) 2) bot))))
                       (t
                        (setq bot
                        (ceiling (+ (/ (+ 1 (- top bot)) 2) bot))))))

                ((eq (aref bits i) #)
                (setq top (floor (+ (/  (- top bot) 2) bot))))
               (t
                (setq bot (ceiling (+ (/ (+ 1 (- top bot)) 2) bot))))))
     bot)
(defun Old-Bits-To-Num (bits top )
;;; Reverse Num-To-Bits.
    (setq bot 0)
    (setq top (- top 1))
        (do ((i 0 (+ 1 i))) ((= top bot))
                (cond ((eq (aref bits i) #)
                (setq top (floor (+ (/  (- top bot) 2) bot))))
               (t
                (setq bot (ceiling (+ (/ (+ 1 (- top bot)) 2) bot))))))
     bot)
(defun Do-Choice (ls)
;;; This is presented by the Do-Op function.
;;; Make a choice and spit it out the right stream.
  (setq temp (Flatten-Choice-List ls))
  (setq tl (Bits-To-Num BitSource (length temp)))
  (cond (debug
         (print (list "In Do Choice with choice:" tl))))
   (write-string
     (nth tl
          temp)
     output-stream))
(defun Do-Reverse-Choice (ls)
;;; Reverse the effects of do-choice.
  (setq answer nil)
  (setq temp (Flatten-Choice-List ls))
  (setq len (length temp))
  (setq counter 0)
  (do ((l temp (cdr l))) ((null l))
      (cond (debug
             (print (list "Checking :" counter (reverse (car l))))))
      (cond ((string= TextSource (reverse (car l))
                      :start1 TextSourcePointer
                      :end1 (+ TextSourcePointer (length (car l))))
             (setq answer
                   (Num-To-Bits counter len))
             (setq TextSourcePointer
                   (+ TextSourcePointer (length (car l))))
             (setq l nil)))
      (setq counter (+ 1 counter)))
```

```
   (cond (debug
          (print (list "Found bits: " (reverse answer)))))
   (setq output-bits (concatenate 'string output-bits (reverse answer)))
   answer)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; Main Code Section
;;;;
;;;;   This is the mastermind.
;;;;   It must open up the correct files and start processing.
;;;;
(defun Encode (In-Data-Name Out-Text-Name &optional (Grammar-File nil))
;;; Encode information so it ends up in a funky grammar file.
        (cond (Grammar-File
               (load Grammar-File))) ;;; If Grammar-File is declared,
                                     ;;; then load more information.
        (setq BitSource (File-To-BitString In-Data-Name))
        (setq output-stream (open Out-Text-Name
                                  :direction :output
                                  :if-does-not-exist :create))
    (setq Forbidden-List nil)
        (Do-Op-List  'main)
        (close output-stream))
(defun Decode (In-Data-Name Out-Text-Name &optional (Grammar-File nil))
;;; Decode information so it comes out in the correct format.
        (cond (Grammar-File
               (load Grammar-File))) ;;; If Grammar-File is declared,
                                     ;;; then load more information.
        (setq TextSource (reverse (File-To-String In-Data-Name)))
        (setq output-bits "")
        (setq TextSourcePointer 0)
        (Do-Reverse-Op-List  'main)
        (BitString-To-File output-bits Out-Text-Name)
)
```

附录D 软　件

很多有用的程序可以在因特网上的存档中找到，其中一些很好的程序如下：

- http://www.stegoarchive.com
- http://crypto.radiusnet.net/archive
- http://www.cl.cam.ac.uk/~fapp2/steganography
- http://www.geocities.com/Paris/9955/priv.html
- http://www.student.seas.gwu.edu/~sowers/digwat.html
- http://www.isse.gmu.edu/~njohnson/Steganography
- http://www.watermarkingworld.org
- http://www.funet.fi/pub/crypt/steganography
- http://glu.freeservers.com/stegano.htm

下面列出在因特网上可利用的专门的软件包：

http://www.stego.com Romana Machado从这里散发出她的Java加密译本和EzStego软件。这个跨越平台的工具是在一个颜色被分类后的图像的最小有效位里隐藏信息的，这样它就通常能工作得很好，但是也可能出现一些不一致性。参看9.2节和17.4节。这个软件是用GNU公共许可来分配的。

http://wwwrn.inf.tu-dresden.de/~westfeld/f5.html 提供了F5软件，它被使用在JPEG图像里隐藏信息，包括也被许多创建者，如Andreas Westfeld[Wes01，WP99]设计用以避免加密技术被发现的提高措施。

http://www.mcdonald.org.uk/StegFS 4.4节里描述的加密文件系统的来源。这个软件使用Linux文件系统，并且可能在某些工作上扩展到任何其他的文件系统中。它在GNU GPL（通用程序设计语言）下被释放。

http://www.smalleranimals.com/stash.htm 用五种不同的技术在图像的最小有效位里隐藏数据的StashIt软件。这个软件是免费的。

http://www.darkside.com.au/snow 由Mattew Kwan开发的Snow（“雪花”干扰效应）软件将在每一个数据行的末尾插入额外的空格，通过加上被包括网络浏览器在内的大多数显示程序忽略的0和7之间的空格，3位就可以在每一数据行里被编码。

http://ftp.csua.berkeley.edu/pub/cryherpunks/steganography/mandelSteg1.0.tor.Z 这个MandelSteg软件在Mandelbrot集合的图像的最小有效位里隐藏信息。这个集合能用7位的精确度合成平面中的任意一组坐标，最后的一位就是信息。

http://www.stella-steganography.de Stella（加密探索实验室）既是在位图里隐藏信息的一个工具又是一个探索可以用什么方法来隐藏信息的实验室。这个软件包括很多不同的分离图像以看到结果的工具。

http://www.darkside.com.au/gifshuffle 由Mattew Kwan编写的GifShuffle程序是根据图像的调色板的顺序来隐藏信息的。如果安排n个对象有n!种不同的方法，那么根据分类的

选择就有$\log_2(n!)$位被隐藏。GifShuffle通过选择256种颜色可隐藏209字节。

http://glu.freeservers.com./sgpo.htm 这个程序 （SteganoGifPaletteOrder）是由David Glaude和Didier Barzin一起编写的。它在GIF调色板中以颜色的排列来隐藏信息，这种方法和GifShuffle程序一样。

http://www.steganos.com Steganos出售一套安全产品，其中包括一个“一个按键单击就消失的硬件驱动”的安全软件。

http://www.tiac.net/users/korejwa/jsteg.htm 用一个Windows命令程序改善了JSteg软件。

http://linux01.gwdg.de/~alatham/stego.html 由Allan Latham编写的JPHide和JPSeek程序使用标准算法在JPEG系数里隐藏信息。这个软件可以保持在系数统计轮廓里的改变轨迹以保证你避免进行密码分析。（参看第17章）

http://www.compris.com/subitext Compris出售的TextHide，一个通过改变句子结构来隐藏信息的软件程序。也就是说，句子被改变以隐藏信息。在理论上，文本在被插入额外的信息后应该还是同样的。

http://www.ctgi.net/nicetext Mark Chapman在Wisconsin（美国威斯康星州）大学学习George Davida的时候，建立了NiceText作为他的标准的论文计划。这个软件集合了一本字典和分类过的字，这样使估计在文本中隐藏信息的形式变得可能[CD97]。

http://www.datamark-tech.com DataMark Techmologies出售使用密码学的四种程序。一种提供了水印技术，一种嵌入未加工的信息，一种给图像加上数字信号，还有一种建立了一个“安全”。

http://www.stealthencrypt.com Stealth Encrypt用它的安全程序组捆绑了一个密码学范例。

http://www.heinz-repp.onlinehome.de/Hide4PGP.htm Hide4PGP在BMP（分类信息程序）或WAV（声音资源）文件的最小有效位里储存数据。它是一个很小的和免费的程序。

http://www.blindside.co.uk Blindside在一个用适当的加密算法作为额外的保护后的位图像里隐藏信息。

http://steghide.sourceforge.net Steghide软件是一个被GPL保护的包，它由Stefan Hetzl创始用来在图像（BMP）或声音（WAV或AU）文件里隐藏信息。

http://www.brasil.terravista.pt/Jenipabu/2571/e-hip.htm 在图像文件的最小有效位里隐藏信息。

http://www.intar.com/ITP/itpinfo.htm 在4-位、8-位和24-位图像中隐藏信息。这个软件也可以用不同的密码储存被保护的复合文件。

http://sourceforge.net/projects/mixmaster Mixmaster是一套运行和使用匿名转信器的很好的工具。

http://idea.sec.dsi.unimi.it/pub/security/crypt/code/s-tools4.zip 这个网址提供了Andrew Brown编写的S-Tools，是在图像和声音文件里隐藏信息的首选程序之一。

http://www.neobytesolution.com/invsecr/index.htm Invisible Secrets是一个在普通位置储存信息的软件。它是一个设计得完美无暇的程序，由标题ads支持的一个译本也是可利用的。

http://www.neobytesolutions.com/invsecr/index.htm S-Mail在x86可执行文件（.exe或.dll）里隐藏信息。这个程序在信息被插入后仍然可以工作。

http://www.camouflagesoftware.co.uk Camouflage是一个压缩、加密以及给一个文件末尾添加信息的基本工具。信息并不被隐蔽地插入到真实的数据中，它只是在末尾被保留。它很棒，并且能保证不给用来掩护的文件带来任何扭曲。

http://wbstego.wbailer.com wbStego是一个在声音、图像和文本格式里隐藏信息的一个完美的专业工具。为了帮助建立所有权，最近的译本也可以在Adobe（美国Adobe公司，是著名的图形图像和排版软件的生产商）的PDF（Adobe的可移植文档格式文件的扩展名）文件里储存信息。

http://www.scramdisk.clara.net 如果你想在硬盘驱动器的一个混杂目录里隐藏信息，Scramdisk就提供了这样的机制。

http://www.cl.cam.ac.uk/~fapp2/steganography/mp3stego Fabien A. P. Petiecolas建立了在现在非常流行的MP3（一种音频压缩格式）文件里隐藏信息的MP3Stego。这个机制使用一个随机数字生成器选择调节一些被数字化的系数的奇偶性[AP98]。

http://www.outguess.org Nile Provos建立了Outguess系统以便不需歪曲统计轮廓就可在JPEG文件里隐藏信息。他也分发了StegDetect程序，它将在另一个加密系统中发现歪曲。

http://www.psionic.com/papers/covert Psionic软件为在冗余的或任意的TCP/IP域（IP包标识领域、TCP初始顺序数字领域和TCP公认顺序数字领域）位里隐藏信息而产生了这个包。

http://ftp.funet.fi/pub/crypt/steganography/PGM.stealth.c.gz 在UNIX操作系统的逻辑单元上，PGMStealth在PGM（Path Generation Method，路经产生方法）文件的最小有效位里隐藏数据。

http://ftp.funet.fi/pub/crypt/steganography/pillo061195.tar.gz 在UNIX（一种多用户的计算机操作系统）逻辑单元上，Piilo在PGM（Path Generation Method，路经产生方法）文件的最小有效位里隐藏数据。

http://www.cl.cam.ac.uk/~fapp2/watermarking/stirmark Stirmark（译者：非常有名的水印测试工具）通过用某种微妙的方法扰乱图像来帮助测试水印和图像加密方法。这个软件像一个橡胶板一样通过展宽一些部分，混乱另一些部分，毁坏一些部分，甚至还复制一些小的部分的方法来处理图像。这被一些参数控制着，所以水印创建者可以这样声称：“这个软件可以技高一筹地抵制Stirmark。”

附录E 更深层次的阅读

本书并不十分完善，因为它只为读者提供了很多课题的一个介绍。因为时间和空间的约束，其他的部分就被省略了。这一部分的目的是为了帮助您更深层次地阅读和探索而提供一些建议。

一个很好的开头之处就是历史记录。David Kahn的“Codebreakers”（电码译员）是一个优秀的密码学历史纵览[Kah67]，有许多像隐显墨水和微粒照片这样的加密解决方法的描述，更近一些的历史记录在“Cruptologia”里被公布。

关于这个主题也有很多其他的好的书。Stefan Katzenbeisser和Fabien Petitcolas从主要的研究者的成果中编辑出了一本评论集*Information Hiding Techniques for Steganography and Digital Watermarking*[SKE00]。Neil F. Johnson，Zoran Duric以及Sushil Jajodia的最近的*Information Hiding: Steganography and Watermarking*是这一系列丛书的第一部分[JDJ01]。Ross Anderson的普通调查*Security Engineering*也包括一些有关隐写术和水印学的信息[And01]。

一些最好的素材可以在它的原始形式里，或在信息隐藏专题讨论会的过程中被找到。现在已经开过四次讨论会，并且更多的讨论会将被列入时间表。

其他更特殊的信息可以在以下这些地方找到。

误码校正代码 本书的第3章还不能充分地评判到很宽的领域，有很多用于不同应用的、不同类型的代码，一些更好的介绍是[LJ83]和[Ara88]，还有很多其他的资料。

压缩算法 压缩始终是一个热门的话题，最近的很多书流行得不太久，最好的解决方法就是把很多书结合在一起，就像[Bar88，BS88]一样，用从比如[Kom95]这样的学术讨论会得来的会议论文集。本书作者最近也写了一本关于压缩的介绍书[Way99]。

阈下通道 这个想法并没有在这本书中被涉及，但是可能很多读者都对它感兴趣。这个领域的大多数工作都是由Gus Simmons来完成的。Gus Simmons发现了很多数字信号算法，这些算法都具有一个能被利用来发送一个额外信息的秘密通道[Sim84，Sim85，Sim86，Sim93，Sim94]。在抽象中理解是相当容易的。很多像El Gamal信号方法[ELG85]或数字信号算法[NCSC93]这样的算法能随机地产生一个数字信号，有很多不同的有效信号存在，算法只随机地取其中一个信号。不用密钥就可产生信号对某些人来说实质上仍然是不可能的，但这些算法的目的只是提供无秘密的鉴定。

设想你如果想给某个人发送一个1位的信息，你能使用的惟一的加密软件是一个DSA信号，而它并不是设计用来隐藏秘密的。你可以发送一个简单巧妙的信息并且重复计算这个信息的数字信号直到最后一位是你的信息位为止。最后，你应该找到通道，因为算法是在信号中随机选择的。

这个抽象的技术只展示了怎样发送一位，还有很多额外的可被利用的有用位，这些评论描述了怎样运用数学以及怎样利用这个通道。

这些算法为关于密码学的政治性的讨论形成了一个重要的例子。美国政府愿意允许人

们使用文电鉴定，但是他们却限制受到秘密保护的加密术的使用，像DSA这样的算法显得非常折衷。尽管如此，阈下通道的存在还是展示了现在流行的算法是怎样的不妥协。

隐蔽的通道 在很多方面"隐蔽的通道"是与本书里使用的同样技术的一个较旧的术语。这个经典的例子来自于操作系统设计：设想你运行一个计算机系统，它有一个应该是安全的操作系统，即意味着OS（译者：Operating System，即操作系统）能够保存两个用户之间传送的信息。很显然，你可以通过关闭如文件复制或电子邮件的服务来实现这样一个操作系统。尽管如此，是否能彻底地消除每一种交流方式还不是很清楚的。

发送一个信息的最简单的例子就是像一个打印机一样占用一些共享资源。如果你想给一个朋友发送一个1，那么你在12:05打印一个文件并且设置好打印机。如果你想发送一个0，那么就在12:30打印文件，另外一个人检查打印机的可用性。这可能不是一个很快的方法，但是它最起码可以工作。通道的速度取决于共享的系统资源和检测的精确性。很显然，抵制秘密通道的一种方法是产生记时错误，但是随之将产生其他的问题。

一些起源的资料在[NCSC93，PN93，MM92]。

数字现金 有很多不同的方法在数字导线上交换钱，但是一些最让人感兴趣的系统提供的是彻底的匿名性，人们能够花他们自己的钱而不怕被保持记录。这是一个相当灵巧的技术，因为数字现金必须是抵制假冒的。当纸币被一个久经考验的技术印刷出来时，它才能够实现抵制假冒的目标。但数字复制则是很容易做到的，如果人们能够复制数字文件来代表现金，那么匿名似乎就允许人们自由地进行假冒伪造而不会被抓着。

最聪明的方法包括一个复杂的支付系统，它强制消费者暴露他或她的部分身份。如果消费者试图使用重复使用一个账单，被暴露的身份就已经足够揭示他的犯罪。

匿名投票 人们通常愿意匿名地投票，因为这可以防止被差强人意。如果没有人在选票进入投票箱之前检查选票，那么纸张投票一般来说是可以成功的（译者：从上下文来看作者的意思可能是说，如果没有人在选票进入投票箱之前检查纸作的选票，那么投票一般来说是可以"成功"的，但不一定公正）。提供同样的责任和安全并不是一门简单的手艺。

例如K. Sako和J. Kilian[SK95]，修改在第10章描述的Mixmaster协议以便为人们提供一个简单的投票方法。每一个人都可以检查记录，并且把被记录的选票和自己的选票相对比以保证选举是公正的，其他的系统包括[BY86，Boy90，FOO93，CC96]。

最后，更新和更好的纸张可以通过像由NEC（http://citeseer.ji.nec.com）运行的CiteSeer系统这样的电子纸张存档被找到。这是一个无价的知识资源。

参考文献

[Age95] National Security Agency. N.S.A. press release: Venona documents released. Technical report, National Security Agency, July 1995.

[AHU83] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, Reading, MA, 1983.

[AK91] Dana Angluin and Michael Kharitonov. When won't membership queries help? In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, pages 444–454. ACM Press, 1991.

[And01] Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley and Sons, New York, 2001.

[ANS98] Ross Anderson, Roger Needham, and Adi Shamir. The steganographic file system. In *IWIH: International Workshop on Information Hiding*, 1998.

[AP98] Ross Anderson and Fabien Petitcolas. The limits of steganography. *IEEE Journal on Selected Areas in Communications*, pages 474–481, May 1998.

[Ara88] Benjamin Arazi. *A Commonsense Approach to the Theory of Error Correcting Codes*. MIT Press, Cambridge, MA, 1988.

[ARC+01] Mikhail Atallah, Victor Raskin, Michael Crogan, Cristian Hempelmann, Florian Kerschbaum, Dina Mohamed, and Sanket Naik. Natural language watermarking: Design, analysis and a proof-of-concept. In *Fourth Information Hiding Workshop*, 2001.

[Are00] S. Areepongsa, N. Kaewkamnerd, Y. F. Syed, and K. R. Rao. Exploring steganography for low bit rate wavelet based coder in image retrieval system. In *Proceedings of TENCOM '00 3*, Kuala Lumpur, Malaysia, September 2000.

[Aur95] Tuomas Aura. Invisible communication. Technical report, Helsinki University of Technology, November 1995.

[Bar88] Michael F. Barnsley. Fractal modelling of real world images. In Heinz-Otto Peitgen and Dietmar Saupe, editors, *The Science of Fractal Images*, Chapter 5, pages 219–239. Springer-Verlag, 1988.

[Bar93] Michael F. Barnsley. *Fractals Everywhere*. 2nd edition. Academic Press, Cambridge, MA, 1993.

[BB00] Mihir Bellare and Alexandra Boldyreva. The security of chaffing and winnowing. In *Lecture Notes in Computer Science, ASIACRYPT*, 1976: 517–530. Springer-Verlag, 2000.

[BBWBG98] S. Blackburn, S. Blake-Wilson, M. Burmester, and S. Galbraith. Shared generation of shared RSA keys. CORR98-19, Department of Combinatorics and Optimization, University of Waterloo, Canada, 1998.

[BF97] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. *Lecture Notes in Computer Science, CRYPTO '97*, 1294: 425–439, 1997.

[BFHMV84] I. F. Blake, R. Fuji-Hara, R. C. Mullin, and S. A. Vanstone. Computing logarithms in finite fields of characteristic two. *SIAM Journal on Algebraic Discrete Methods*, 5, 1984.

[BGML96] Walter Bender, D. Gruhl, N. Morimoto, and A. Lu. Techniques for data hiding. *IBM Systems Journal*, 35(3): 313. 1996.

[BH92] M. Barnsley and L. Hurd. *Fractal Image Compression*. AK Peters, Ltd., Wellesley, MA, 1992.

[BJNW57] F. P. Brooks, A. L. Hopkins Jr., Peter G. Neumann, and W. V. Wright. An experiment in musical composition. *IRE Transactions on Electronic Computers*. EC-6(3), September 1957.

[BL85] Charles Bennett and Rolf Landauer. The fundamental physical limits of computation. *Scientific American*, pages 48–56, July 1985.

[BLMO94] J. Brassil, S. Low, N. Maxemchuk, and L. O'Gorman. Electronic marking and identification techniques to discourage document copying. In *Proceedings of IEEE Infocom 94*, pages 1278–1287, 1994.

[BLMO95] Jack Brassil, Steve Low, Nicholas Maxemchuk, and Larry O'Gorman. Hiding information in document images. In *Proceedings of the 1995 Conference on Information Sciences and Systems*, March 1995.

[Blu82] M. Blum. Coin flipping by telephone: A protocol for solving impossible problems. *Proceedings of the 24th IEEE Computer Conference (CompCon)*, 1982.

[BMS01] Adam Back, Ulf Moeller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Fourth Information Hiding Workshop*, pages 257–269, 2001.

[BO96] Jack Brassil and Larry O'Gorman, Watermarking document images with bounding box expansion. In *Information Hiding, Lecture Notes of Computer Science (1174)*. Springer-Verlag, New York, Heidelberg, 1996.

[Boy90] C. Boyd. A new multiple key cipher and an improved voting scheme. In *Advances in Cryptology—EUROCRYPT '89 Proceedings*. Springer-Verlag, 1990.

[Bra93] S. A. Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CSR9323, Computer Science Department, Centrum voor Wiskunde en Informatica, Amsterdam, March 1993.

[Bra95a] Stefan A. Brands. *Rethinking Public Key Infrastructure and Digital Certificates—Building in Privacy*. Ph.D. thesis, Amsterdam, 1995.

[Bra95b] Stefan A. Brands. Secret-key certificates. Technical Report CS-R9510, Centrum voor Wiskunde en Informatica, Amsterdam, 1995.

[Bri82] E. F. Brickell. A fast modular multiplication algorithm with applications to two key cryptography. In *Advances in Cryptology: Proceedings of Crypto 82*. Plenum Press, 1982.

[BS88] Michael F. Barnsley and Alan D. Sloan. A better way to compress images. *Byte Magazine*, pages 215–223, January 1988.

[BS95] D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. In *15th Annual International Cryptology Conference*, number 963, pages 452–465, Santa Barbara, CA, 1995.

[BS99] R. W. Buccigrossi and E. P. Simoncelli. Image compression via joint statistical characterization in the wavelet domain. *IEEE Transactions on Image Processing*, 8(12): 1688–1701. 1999.

[BY86] J. C. Benaloh and M. Yung. Distributing the power of government to enhance the privacy of voters. *Proceedings of the 5th ACM Symposium on the Principles in Distributed Computing*, 1986.

[CC96] Lorrie Cranor and R. Cytron. Design and implementation of a practical security-conscious electronic polling system. Technical Report WUCS-96-02, Washington University Department of Computer Science, St. Louis, 1996.

[CD97] Mark Chapman and George Davida. Hiding the hidden: A software system for concealing ciphertext as innocuous text. In *International Conference on Information and Computer Security (ICICS '97)*, Beijing, P.R. China, November 1997.

[Cha81] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2), February 1981.

[Cha95a] David Charlap. The BMP file format, Part i. *Dr. Dobbs Journal*, March 1995.

[Cha95b] David Charlap. The BMP file format, Part ii. *Dr. Dobbs Journal*, April 1995.

[CKLS96] Ingemar Cox, Joe Kilian, Tom Leighton, and Talal Shamoon. A secure, robust watermark for multimedia. In *Information Hiding, Lecture Notes of Computer Science (1174)*. Springer-Verlag, New York, Heidelberg, 1996.

[Cla83] F. Clarke. *Optimization and Nonsmooth Analysis*. John Wiley, New York, 1983.

[Cla99] Ian Clarke. A distributed decentralised information storage and retrieval system. Department of Combinatorics and Optimization, University of Waterloo, 1999.

[CM58] Noam Chomsky and G. A. Miller. Finite state languages. *Information and Control*, 1: 91–112. 1958.

[CM97] Ingemar J. Cox and Matt L. Miller. A review of watermarking and the importance of perceptual modeling. In *Proceedings of Electronic Imaging '97*, February 1997.

[CM98] J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In *ASIACRYPT: Advances in Cryptology—ASIACRYPT: International Conference on the Theory and Application of Cryptology*. LNCS, Springer-Verlag, 1998.

[CMB01] Ingemar J. Cox, Matthew L. Miller, and Jeffrey A. Bloom. *Digital Watermarking*. Morgan Kaufmann, San Fransisco, CA, 2001.

[CSWH00] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, CA, 2000.

[CW93] K. W. Campbell and M. J. Wiener. DES is not a group. In *Advances in Cryptology—CRYPTO '92 Proceedings*. Springer-Verlag, 1993.

[DF00] D. M. Roger Dingledine and Michael J. Freedman. The Free Haven project: Distributed anonymous storage service. In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, July 2000.

[DR00] Joan Daemen and Vincent Rijmen. The block cipher Rijndael, pages 288–296. Springer-Verlag, 2000.

[DR01] Joan Daemen and Vincent Rijmen. Rijndael, the advanced encryption standard. *Dr. Dobb's Journal*, 26(3): 137–139, March 2001.

[DIF76a] W. Diffie and M. E. Hellman. Multiuser cryptographic techniques. In *Proceedings of AFIPS National Computer Conference*, 1976.

[DIF76b] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6), November 1976.

[ElG85] T. El Gamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology: Proceedings of CRYPTO 84*. Springer-Verlag, 1985.

[ESG00a] Joachim J. Eggers, Jonathan K. Su, and Bernd Girod. Asymmetric watermarking schemes. In *Sicherheit in Mediendaten*, Berlin, September 2000.

[ESG00b] Joachim J. Eggers, Jonathan K. Su, and Bernd Girod. Public-key watermarking by eigenvectors of linear transforms. In *EUSIPCO 2000*, Tampere, Finland, September 2000.

[Ett98] J. Mark Ettinger. Steganalysis and game equilibria. In *Information Hiding Workshop, Lecture Notes of Computer Science (1525)*. Springer-Verlag, New York, Heidelberg, 1998.

[FBS96] J. Fridrich, Arnold Baldoza, and Richard Simard. Robust digital watermarking based on key-dependent basis functions. In *Information Hiding Workshop, Lecture Notes of Computer Science (1525)*. Springer-Verlag, New York, Heidelberg, 1996.

[FDL] J. Fridrich, Rui Du, and Meng Long. Steganalysis of LSB encoding in color images. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, August 2000.

[FG99] J. Fridrich and M. Goljan. Protection of digital images using self embedding. In *Proceedings of NJIT Symposium on Content Security and Data Hiding in Digital Media*, Newark, NJ, May 1999.

[FMY98] Yair Frankel, Philip D. MacKenzie, and Moti Yung. Robust efficient distributed RSA-key generation. In *Symposium on Principles of Distributed Computing*, page 320, 1998.

[FOO93] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology—AUSCRYPT '92 Proceedings*. Springer-Verlag, 1993.

[Fou98] Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics and Chip Design*. O'Reilly, 1998.

[Fre82] Ed Fredkin. Conservative logic. *International Journal of Theoretical Physics*, 21, 1982.

[Fri99] J. Fridrich. A new steganographic method for palette-based images. In *Proceedings of the IS&T PICS Conference*, Savannah, GA, April 1999.

[FT82] Edward Fredkin and Tommaso Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21: 219–253. 1982.

[GB98] Daniel Gruhl and Walter Bender. Information hiding to foil the casual counterfeiter. In *Information Hiding Workshop, Lecture Notes of Computer Science (1525)*. Springer-Verlag, New York, Heidelberg, 1998.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[GJKR] R. Gennaro, T. Rabin, S. Jarecki, and H. Krawczyk. Robust and efficient sharing of RSA functions. *Journal of Cryptology: The Journal of the International Association for Cryptologic Research*, 13(2): 273–300. 2000.

[Gro] Wendy Grossman. alt.scientology.war. *Wired*, 3(2): 172, December 1995.

[Gun88] C. G. Gunther. A universal algorithm for homophonic coding. In *Advances in Cryptology—Eurocrypt '88 Lecture Notes in Computer Science* (330), pages 405–414. Springer-Verlag, New York, 1988.

[Hec82] Paul Heckbert. Color image quantization for frame buffer display. In *Proceedings of SIGGRAPH 82*, 1982.

[HG] Frank Hartung and Bernd Girod. Fast public-key watermarking of compressed video. In *Proceedings of International Conference on Image Processing (ICIP '97) 1*, Santa Barbara, CA, 1997.

[Hil91] David Hillman. The structure of reversible one-dimensional cellular automata. *Physica D*, 54: 277–292, 1991.

[HSG99] Frank Hartung, Jonathan K. Su, and Bernd Girod. Spread spectrum watermarking: Malicious attacks and counterattacks. In *Security and Watermarking of Multimedia Contents, Proceedings for SPIE (The International Society for Optical Engineering) (3657)*, pages 147–158, January 1999.

[HU79] John Hopcroft and Jeffrey Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, MA, 1979.

[JDJ01] Neil F. Johnson, Zoran Duric, and Sushil Jajodia. *Information Hiding: Steganography and Watermarking—Attacks and Countermeasures (Advances in Information Security, Volume 1)*. Kluwer Academic Publishers, 2001.

[JJ98a] Neil F. Johnson and Sushil Jajodia. Steganalysis of images created using current steganography software. In *Information Hiding, Second International Workshop*, pages 273–289, 1998.

[JJ98b] Neil F. Johnson and Sushil Jajodia. *Steganalysis: The investigation of hidden information.* 1998.

[JKM90] H. N. Jendal, Y. J. B. Kuhn, and J. L. Massey. An information-theoretic treatment of homophonic substitution. In *Advances in Cryptology—Eurocrypt '89, Lecture Notes of Computer Science.* Springer-Verlag, New York, 1990.

[KA98] Markus G. Kuhn and Ross J. Anderson. Soft tempest: Hidden data transmission using electromagnetic emanations. In *Information Hiding Workshop, Lecture Notes of Computer Science (1525).* Springer-Verlag, New York, Heidelberg, 1998.

[Kah67] David Kahn. *The Codebreakers.* Macmillan, New York, 1967.

[KBC+00] John Kubiatowicz, David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westly Weimer, Christopher Wells, and Ben Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS.* ACM, 2000.

[Kea89] Michael Kearns. *The Computational Complexity of Machine Learning.* Ph.D. thesis, Harvard University Center for Research in Computing Technology, May 1989.

[KH98] Deepa Kundur and Dimitrios Hatzinakos. Digital watermarking using multiresolution wavelet decomposition. In *International Conference on Acoustic, Speech and Signal Processing (ICASP),* volume 5, pages 2969–2972. 1998.

[KH99] Deepa Kundur and D. Hatzinakos. Digital watermarking for telltale tamper-proofing and authentication. *Proceedings of the IEEE Special Issue on Identification and Protection of Multimedia Information,* 87(7): 1167–1180, July 1999.

[Knu81] D. Knuth. *The Art of Computer Programming: Volume 2, Seminumerical Algorithms.* 2nd edition. Addison-Wesley, Reading, MA, 1981.

[KO84] Hugh Kenner and Joseph O'Rourke. A travesty generator for micros. *BYTE,* page 129, November 1984.

[Kob87] Neal Koblitz. *A Course in Number Theory and Cryptography.* Springer-Verlag, New York, 1987.

[Kom95] John Kominek. Convergence of fractal encoded images. In J. A. Storer and M. Cohn, editors, *Data Compression Conference 1995,* pages 242–251, Snowbird, UT, March 1995.

[KQP01] Farinaz Koushanfar, Gang Qu, and Miodrag Potkonjak. Intellectual property metering. In *Fourth Information Hiding Workshop,* 2001.

[KV89] Michael Kearns and Leslie Vallent. Cryptographic limitations on learning boolean formulae and finite automata. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 433–444, Seattle, May 1989.

[Leh82] D. J. Lehmann. On primality tests. *SIAM Journal on Computing*, 11(2), May 1982.

[Lia95] Wilson MacGyver Liaw. Reading GIF files. *Dr. Dobbs Journal*, February 1995.

[Lic] Vinicius Licks and R. Jordan. On digital image watermarking robust to geometric transformations. In *Proceedings of IEEE International Conference on Image Processing 2000*, Vancouver, Canada, 2000.

[LJ83] Shu Lin and Daniel J. Costello Jr. *Error Control Coding: Fundaments and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1983.

[LMBO95] Steve Low, Nicholas Maxemchuk, Jack Brassil, and Larry O'Gorman. Document marking and identification using both line and word shifting. In *Proceedings of the 1995 Conference on Infocom '95*, April 1995.

[LMSP98] John Lach, William H. Mangione-Smith, and Miodrag Potkonjak. Fingerprinting digital circuits on programmable hardware. In *Information Hiding Workshop, Lecture Notes of Computer Science (1525)*. Springer-Verlag, New York, Heidelberg, 1998.

[Mae98] Maurise Maes. Twin peaks: The histogram attack to fixed depth image watermarks. In *Information Hiding Workshop, Lecture Notes of Computer Science (1525)*. Springer-Verlag, New York, Heidelberg, 1998.

[Mar84] N. Margolus. Physics-like models of computation. *Physica D*, 10: 81–95, 1984.

[MBR99] L. Marvel, C. Boncelet, and J. Retter. Spread spectrum image steganography. *IEEE Transactions on Image Processing*, 8(8): 1075–1083. 1999.

[McH01] John McHugh. Cover image. In *Fourth Information Hiding Workshop*. 2001.

[Mer93] Ralph Merkle. Reversible electronic logic using switches. *Nanotechnology*, 4: 21–40. 1993.

[MM92] I. S. Moskowitz and A. R. Miller. The channel capacity of a certain noisy timing channel. *IEEE Transactions on Information Theory*, IT-38(4): 1339–1343. 1992.

[Mon85] P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44: 170. 1985.

[MWC00] Aviel D. Rubin, Marc Waldman, and Lorrie Faith Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proceedings of the 9th USENIX Security Symposium*, pages 59–72, August 2000.

[NCSC93] A guide to understanding covert channel analysis of trusted systems. Technical Report TG-030, NCSC, November 1993.

[Neu64] P. G. Neumann. Error-limiting coding using information-lossless sequential machines. *IEEE Transactions on Information Theory*, IT-10: 108–115, April 1964.

[NY89] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 33–43. ACM, 1989.

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 427–437. ACM, 1990.

[PBBC97] A. Piva, M. Barni, F. Bartolini, and V. Cappellini. DCT-based watermark recovering without resorting to the uncorrupted original image. In *IEEE Signal Processing Society 1997 International Conference on Image Processing (ICIP '97)*, Santa Barbara, CA, October 1997.

[PN93] N. Proctor and P. Neumann. Architectural implications of covert channels. In *Proceedings of the 15th National Computer Security Conference*, 1993.

[PP90] B. Pfitzmann and A. Pfitzmann. How to break the direct RSA-implementation of mixes. In *Advances in Cryptology—Eurocrypt '89*, number 434. Springer-Verlag, 1990.

[Pro] Niels Provos. Defending against statistical steganalysis. In *Proceedings of the 10th USENIX Security Symposium*, pages 323–335. 2001.

[Pro01] Niels Provos. Probabilistic methods for improving information hiding. Technical Report 01-1, University of Michigan, January 2001.

[QC82] J.-J. Quisquater and C. Couvreur. Fast decipherment algorithm for RSA public-key cryptosystem. *Electronic Letters*, 18. 1982.

[Qu01] Gang Qu. Keyless public watermarking for intellectual property authentication. In *Fourth Information Hiding Workshop*. 2001.

[Rab89] Michael Rabin. Efficient dispersal of information for security, load balancing and fault tolerance. *Journal of the ACM*, 38: 335–348. 1989.

[RC95] R. Rinaldo and G. Calvagno. Image coding by block prediction of multiresolution subimages. *IEEE Transactions on Image Processing*, 4: 909–920. 1995.

[RDB96] J. Ruanaidh, W. Dowling, and F. Boland. Phase watermarking of digital images. In *Proceedings of ICIP '96*, 3: 239–242, Lausanne, Switzerland, September 1996.

[Riv] Ron Rivest. Chaffing and winnowing: Confidentiality without encryption. *Crypto Bytes* (RSA Laboratories), 4(1): 12–17, Summer 1998.

[Rob62] L. G. Roberts. Picture coding using pseudo-random noise. *IRE Transactions on Information Theory*, IT-8, February 1962.

[RR98] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1): 66–92. 1998.

[RSG98] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4): 482–494, May 1998.

[Sch94] Bruce Schneier. *Applied Cryptography*. John Wiley and Sons, New York, 1994.

[SGR97] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *Proceedings/1997 IEEE Symposium on Security and Privacy, May 4–7, 1997, Oakland, California*, pages 44–54. IEEE Computer Society Press, Silver Springs, MD, 1997.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 24(11), November 1979.

[Sha93] J. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12): 3445–3462. 1993.

[Sha01] Toby Sharp. An implementation of key-based digital signal steganography. In *Fourth Information Hiding Workshop*, 2001.

[Shi99] Natori Shin. One-time hash steganography. In *3rd Information Hiding Workshop, Lecture Notes of Computer Science (1768)*. Springer-Verlag, New York, Heidelberg, 1999.

[Sim84] G. J. Simmons. The prisoner's problem and the subliminal channel. In *Advances in Cryptology: Proceedings of CRYPTO '83*. Plenum Press, 1984.

[Sim85] G. J. Simmons. The subliminal channel and digital signatures. In *Advances in Cryptology: Proceedings of EUROCRYPT 84*. Springer-Verlag, 1985.

[Sim86] G. J. Simmons. A secure subliminal channel. In *Advances in Cryptology—CRYPTO '85 Proceedings*. Springer-Verlag, 1986.

[Sim92] G. J. Simmons, ed. *Contemporary Cryptology: The Science of Information Integrity*. IEEE Press, Piscataway, NJ, 1992.

[Sim93] G. J. Simmons. The subliminal channels of the U.S. Digital Signature Algorithm (DSA). In *Proceedings of the Third Symposium on State and Progress of Research in Cryptography*, Fondazone Ugo Bordoni, Rome, 1993.

[Sim94] G. J. Simmons. Subliminal communication is easy using the DSA. In *Advances in Cryptology—EUROCRYPT '93 Proceedings*. Springer-Verlag, 1994.

[SK95] K. Sako and J. Kilian. Receipt-free mix-type voting schemes. In *Advances in Cryptology—Eurocrypt '95*, pages 393–403. Springer-Verlag, 1995.

[SKE00] Fabien Petitcolas and Stefan Katzenbeisser (Editors). *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House. 2000.

[SRG00] Paul F. Syverson, Michael G. Reed, and David M. Goldschlag. Onion routing access configurations. In *DISCEX 2000: Proceedings of the DARPA Information Survivability Conference and Exposition*, Volume I, pages 34–40, Hilton Head, SC, January 2000. IEEE CS Press.

[Sto88] James Storer. *Data Compression*. Computer Science Press, Rockville, MD, 1988.

[STRL00] Paul F. Syverson, Gene Tsudik, Michael G. Reed, and Carl E. Landwehr. Towards an analysis of onion routing security. In *Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, CA, July 2000.

[SY98] Sabrina Sowers and Abdou Youssef. Testing digital watermark resistance to destruction. In *Information Hiding Workshop, Lecture Notes of Computer Science (1525)*. Springer-Verlag, New York, Heidelberg, 1998.

[Tah92] H. Taha. Operations Research: An Introduction. Macmillan, New York, 1992.

[TM87] Tommaso Toffoli and Norman Margolus. *Cellular Automata Machines*. MIT Press, London, 1987.

[Tof77a] T. Toffoli. *Cellular Automata Mechanics*. Ph.D. thesis, University of Michigan, 1977.

[Tof77b] Tommaso Toffoli. Computation and construction universality of reversible cellular automata. *Journal of Computer and System Sciences*, 15: 213–231. 1977.

[Tur36a] Alan Turing. On computable numbers with an application to the entscheidungsproblem. *Proceedings of the London Math Society*, 2(42): 230–265. 1936.

[Tur36b] Alan Turing. On computable numbers with an application to the entscheidungsproblem. *Proceedings of the London Math Society*, 2(43): 544–546. 1936.

[Val84] Leslie G. Valient. A theory of the learnable. *Communications of the ACM*, 27: 1134–1142. 1984.

[Wal95] S. Walton. Image authentication for a slippery new age. *Dr. Dobbs Journal*, 20(4): 18–26, April 1995.

[Way85] Peter Wayner. Building a travesty tree. *BYTE*, page 183, September 1985.

[Way92] Peter C. Wayner. Content-addressable search engines and DES-like systems. In *Advances in Cryptology: CRYPTO '92 Lecture Notes of Computer Science (740)*, pages 575–586. Springer-Verlag, New York, 1992.

[Way95] Peter Wayner. Strong theoretical steganography. *Cryptologia*, 19(3): 285–299, July 1995.

[Way99] Peter Wayner. *Data Compression for Real Programmers*. AP Professional, Chesnut Hill, MA, 1999.

[Wei76] Joseph Weizenbaum. *Computer Power and Human Reason: From Judgment to Calculation*. W. H. Freeman, San Francisco, 1976.

[Wes01] Andreas Westfeld. High capacity despite better steganalysis: F5, a steganographic algorithm. In *Fourth Information Hiding Workshop*, pages 301–315. 2001.

[WH94] Peter Wayner and Dan Huttenlocher. Image analysis to obtain typeface information. *U.S. Patent*, 5253307, 1994.

[Won98] P. Wong. A public key watermark for image verification and authentication. In *Proceedings of ICIP '98 1*, pages 425–429, Chicago, IL, October 1998.

[WP99] Andreas Westfeld and Andreas Pfitzmann. Attacks on steganographic systems. In *Information Hiding, Third International Workshop, IH '99* (1768), pages 61–76. Springer-Verlag, Dresden, Germany, 1999.

[WRC00] Marc Waldman, Aviel D. Rubin, and Lorrie Faith Cranor. Publins: A robust, tamper-evident, censorship-resistant, Web publishing system. In *Proceedings of 9th Security Symposium*, pages 59–72, August 2000.

[WS99] Wright and Spalding. Experimental performance of shared RSA modulus generation (short). In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1999.

[XBA97] X.-G. Xia, C. Boncelet, and G. Arce. A multiresolution watermark for digital images. In *IEEE Signal Processing Society 1997 International Conference on Image Processing (ICIP '97)*, Santa Barbara, CA, October 1997.

欢迎与我们联系

为了方便与我们联系，我们已开通了网站（www.medias.com.cn）。您可以在本网站上了解我们的新书介绍，并可通过读者留言簿直接与我们沟通，欢迎您向我们提出您的想法和建议。